STIC Search Report
EIC 2100

STIC Database Tracking Number: 165915

TO:	From:
Susan F Rayyan Location: RND 3C05 Art Unit : 2167 Thursday, June 09, 2005 Case Serial Number: 10/023433	David Holloway Location: EIC 2100 RND 4B19 Phone: 2-3528 david.holloway@uspto.gov

Search Notes

Dear Examiner Rayyan,

Attached please find your search results for above-referenced case.
Please contact me if you have any questions or would like a re-focused search.

David

155915

(34)

STIC EIC 2100
Search Request Form

Today's Date: 6/9/05

What date would you like to use to limit the search?
Priority Date: 12/17/07 Other:

Name Susan Ruyan
AU 2167 Examiner # 77889
Room # 3C05 Phone 24117
Serial # 10/023433

Format for Search Results (Circle One):
(PAPER) DISK EMAIL

Where have you searched so far?
USP DWPI EPO JPO ACM IBM TDB
IEEE INSPEC SPI Other _____

Is this a "Fast & Focused" Search Request? (Circle One) (YES) NO
A "Fast & Focused" Search is completed in 2-3 hours (maximum). The search must be on a very specific topic and meet certain criteria. The criteria are posted in EIC2100 and on the EIC2100 NPL Web Page at http://ptoweb/patents/stic/stic-tc2100.htm.

What is the topic, novelty, motivation, utility, or other specific details defining the desired focus of this search? Please include the concepts, synonyms, keywords, acronyms, definitions, strategies, and anything else that helps to describe the topic. Please attach a copy of the abstract, background, brief summary, pertinent claims and any citations of relevant art you have found.

Title Text search ordered along one or more Dimensions.
User submits query performing search for documents using different search criteria including different dimensions

search criteria ordered based

see spec. p. 16
Dimension
search order
Search criteria

Query class = # search terms in query.

dimensions
Text Dimension
measures
~~linguist~~
portions of documents (Title abstract) Titles
linguistic Dimensions

Examiner will explain.

STIC Searcher David Hilliary Phone 2-3528
Date picked up 6-9-05 Date Completed 6-9-05

INTERNET ARCHIVE
WaybackMachine

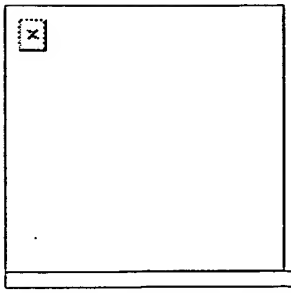
Enter Web Address: http:// All Take Me Back Adv. Search Compare Arch

Searched for **http://dogpile.com** **1767** Results

Note some duplicates are not shown. See all.
* denotes when site was updated.

Search Results for Jan 01, 1996 - Jun 09, 2005

1996	1997	1998	1999	2000	2001	2002
1 pages	5 pages	3 pages	16 pages	100 pages	1285 pages	121 pages
Dec 20, 1996 *	Apr 12, 1997 *	Jan 15, 1998 *	Jan 16, 1999 *	Feb 29, 2000 *	Jan 04, 2001 *	Jan 22, 2002 *
	Apr 19, 1997 *	Feb 04, 1998 *	Jan 17, 1999	Feb 29, 2000 *	Jan 04, 2001 *	Jan 24, 2002
	Jun 06, 1997 *	Dec 12, 1998 *	Jan 25, 1999	Feb 29, 2000 *	Jan 05, 2001 *	Jan 24, 2002 *
	Oct 18, 1997 *		Feb 08, 1999 *	Feb 29, 2000 *	Jan 05, 2001 *	Jun 02, 2002 *
	Dec 10, 1997 *		Feb 09, 1999	Feb 29, 2000 *	Jan 06, 2001 *	Jun 03, 2002 *
			Feb 20, 1999	Feb 29, 2000 *	Jan 06, 2001 *	Jun 04, 2002 *
			Apr 20, 1999 *	Feb 29, 2000 *	Jan 06, 2001 *	Jul 19, 2002 *
			Apr 23, 1999	Mar 01, 2000 *	Jan 07, 2001 *	Aug 02, 2002 *
			Apr 29, 1999 *	Mar 01, 2000 *	Jan 08, 2001 *	Aug 02, 2002 *
			May 02, 1999	Mar 01, 2000 *	Jan 08, 2001 *	Aug 04, 2002 *
			Oct 07, 1999 *	Mar 01, 2000 *	Jan 18, 2001 *	Aug 06, 2002 *
			Oct 12, 1999 *	Mar 01, 2000 *	Jan 19, 2001 *	Aug 07, 2002 *
			Oct 13, 1999 *	Mar 02, 2000 *	Jan 19, 2001 *	Aug 07, 2002 *
			Oct 13, 1999 *	Mar 02, 2000 *	Jan 19, 2001 *	Aug 08, 2002 *
			Nov 04, 1999 *	Mar 02, 2000 *	Jan 19, 2001 *	Aug 09, 2002 *
			Nov 28, 1999 *	Mar 03, 2000 *	Jan 19, 2001 *	Aug 10, 2002 *
				Mar 03, 2000 *	Jan 19, 2001 *	Aug 11, 2002
				Mar 03, 2000 *	Jan 19, 2001 *	Aug 12, 2002 *
				Mar 03, 2000 *	Jan 19, 2001 *	Aug 12, 2002 *
				Mar 03, 2000 *	Jan 19, 2001 *	Aug 13, 2002
				Mar 03, 2000 *	Jan 19, 2001 *	Aug 13, 2002 *
				Apr 07, 2000 *	Jan 19, 2001 *	Aug 14, 2002 *
				Apr 07, 2000 *	Jan 19, 2001 *	Aug 15, 2002
				May 10, 2000 *	Jan 19, 2001 *	Aug 16, 2002
				May 10, 2000 *	Jan 19, 2001 *	Aug 18, 2002
				May 10, 2000 *	Jan 19, 2001 *	Aug 19, 2002
				May 10, 2000 *	Jan 19, 2001 *	Aug 21, 2002 *
				May 10, 2000 *	Jan 19, 2001 *	Aug 23, 2002 *
				May 10, 2000 *	Jan 19, 2001 *	Aug 24, 2002
				May 10, 2000 *	Jan 19, 2001 *	Aug 25, 2002
				May 10, 2000 *	Jan 19, 2001 *	Aug 26, 2002 *
				May 10, 2000 *	Jan 19, 2001 *	Aug 27, 2002 *
				May 10, 2000 *	Jan 24, 2001 *	Aug 28, 2002
				May 10, 2000 *	Jan 30, 2001 *	Aug 29, 2002
				May 10, 2000 *	Mar 02, 2001 *	Aug 30, 2002
				May 10, 2000 *	Mar 02, 2001 *	Aug 31, 2002
				May 10, 2000 *	Mar 02, 2001 *	Sep 01, 2002 *
				May 11, 2000 *	Mar 02, 2001 *	Sep 02, 2002 *
				May 11, 2000 *	Mar 02, 2001 *	Sep 03, 2002 *
				May 11, 2000 *	Mar 31, 2001 *	Sep 05, 2002
				May 11, 2000 *	Apr 05, 2001 *	Sep 07, 2002



DOGPILE

Arfie... our very happy now wagging his tail mascot.

Welcome to The Dogpile Search Gateway
We Found A New Home!!.

Because we are changing our location on Saturday and because Internic records are being changed tonight or tomorrow night, It is likely that dogpile will be unreachable for some unknown amount of time between tonight and Monday. Everything should be running on Monday. Wish me luck.

Dogpile Remote Help on syntax About Us Technical Notes Known Bugs and Problems

Try **Arfie!** Arfie is now running on our new machine, via an ISDN line. The new connection should be up in about a week. Also, I may need to switch back to our slow machine from time to time.

Search The Web and then STOP Fetch
Wait a maximum of Forty Seconds.

Searches:
The Web: *Yahoo!, Lycos' A2Z, World Wide Web Worm, Yahoo's new search, What U Seek, Lycos, WebCrawler, InfoSeek, OpenText, AltaVista, Excite & HotBot.*
Usenet: *Reference.com, Dejanews, Altavista and Dejanews' old Database.*
FTP: *FTP Search and Snoopie!.*(Only the first word in your query will be passed on to these search engines.)

Let us know what you think! Comments and suggestions are welcome.

Custom Search
Dogpile Remote
Stock Quotes
Business News
Search at Home
Help with Syntax

DOGPILE: A MultiSearch Engine

MetaFind Search
Weather
Feedback
Advertising Info

Help with Syntax

The **Dogpile** search interface takes a single line for a query and processes it so that you will get the maximum benefit from your search. Currently twenty five search engines are supported. They are:

- WWW: Yahoo, Lycos' A2Z, Go2.com, Excite Guide Search, WWW Yellow Pages, Thunderstone, What U Seek, Lycos, PlanetSearch, Magellan, WebCrawler, InfoSeek, AltaVista & Excite.
- Usenet: Reference.com, Dejanews, Altavista and Dejanews' old Database.
- FTP: Filez and FTP Search.
- News Wires: Yahoo News Headlines, Excite News, Infoseek News Wires.

What is Allowed?

- You may use the proximity and Boolean operators **AND**, **OR**, **NEAR**, and **NOT** to combine words and phrases. **NEAR** will be substituted with **AND** for those engines which do not support its use. If you use **NEAR** the engines which support its use will be searched first. **NOT** and the following word will be deleted if the engine does not support its use.
- **OR** is not fully supported since not all the search engines included support a mixed use of **AND** and **OR**. This is not a limitation for MetaFind however.
- Using no connector, **AND** will be assumed. Thus the search:
  - Free and Mac and Software

  and
  - Free Mac Software

  are the same.

- You may also use quotes and parentheses. However note that not all search engines support their use. For those which do not support their use, they will *automatically* be removed.

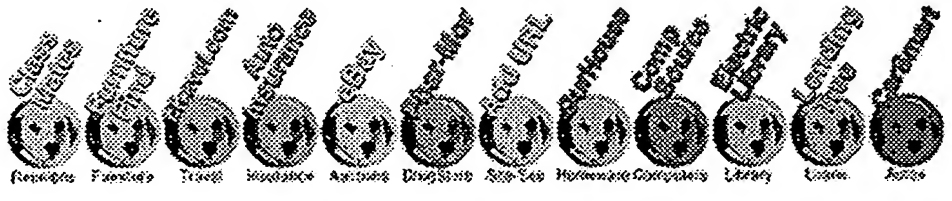
- The FTP search engines only take one word as the query (i.e. a file name or part of a file name) Make sure that the first word in

your query is the file name you want to find. You may still add other search terms if you also are searching the web or usenet also.

What will Happen when I Press "Fetch"

- **Arfie** searches three search engines at a time. The requests are put out in parallel and are displayed as they come back.
- If **Arfie** does not get at least 10 documents matching your query request it will automatically move to the next three and so on until all are searched or until 10 matches are found.
- If **Arfie** does find 10 or more matches you still can go to the next set of search engines by pressing on the button at the bottom of the page.
- Engines have generally been placed in order from the very general index search (where a general search like "usenet culture" will not turn up 30000 pages) to the very specific super-engine (which will find too much if your search is not narrow). This means that you can put in as much or as little detail about what it is you want to find and not be disappointed with too few matches (since Arfie automatically fetches more documents from larger databases if less than ten are found) nor be overwhelmed when looking for information on a general topic (since the index search engines like Yahoo are catagorized into general subject headings).
- Attempts have been made to make it easy to follow up on the query sent to each search engine. The query sent to the search engine is printed and (in most cases) is linked to the page generating the results. Also if the search engine found more than the maximum displayable matches (e.g. 10 matches), a link to the next 10 should also be present (where supported).
- **Please remember that search engines change their format all the time. Thus Arfie is guaranteed NOT to work 100% of the time with 100% of the engines.**
- **If you still do not understand any of this, don't worry. Just try it and see what happens.**

Search The Web and then STOP Fetch
Wait a maximum of Forty Seconds.



Title: **Better Web searches and prediction with instantaneously trained neural networks.**

Authors: Kak, Subhash

Source: IEEE Expert Intelligent Systems & Their Applications; Nov/Dec99, Vol. 14 Issue 6, p78, 4p, 2 charts, 2 graphs, 1bw

Document Type: Article

Subject Terms: *NEURAL networks (Computer science)
*WEB search engines

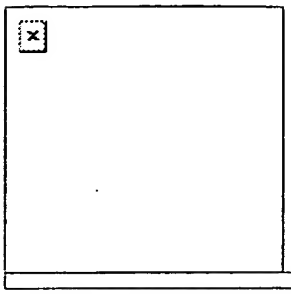
Abstract: Explores better Web searches and prediction with instantaneously trained neural networks. Different kinds of memory; Description of working memory; Model for instantaneously learned working memory; Forecasting in a time series; Design of an intelligent metasearch engine.

ISSN: 0885-9000

Accession Number: 9591156

Persistent link to this record: http://search.epnet.com/login.aspx?direct=true&db=aph&an=9591156

Database: Academic Search Premier



DOGPILE

Arfie... our very happy now wagging his tail mascot.

Welcome to The Dogpile Search Gateway
We Found A New Home!!.

Because we are changing our location on Saturday and because Internic records are being changed tonight or tomorrow night, It is likely that dogpile will be unreachable for some unknown amount of time between tonight and Monday. Everything should be running on Monday. Wish me luck.

Dogpile Remote Help on syntax About Us Technical Notes Known Bugs and Problems

Try **Arfie!** Arfie is now running on our new machine, via an ISDN line. The new connection should be up in about a week. Also, I may need to switch back to our slow machine from time to time.

Search The Web and then STOP Fetch
Wait a maximum of Forty Seconds.

Searches:
The Web: *Yahoo!, Lycos' A2Z, World Wide Web Worm, Yahoo's new search, What U Seek, Lycos, WebCrawler, InfoSeek, OpenText, AltaVista, Excite & HotBot.*
Usenet: *Reference.com, Dejanews, Altavista and Dejanews' old Database.*
FTP: *FTP Search and Snoopie!.*(Only the first word in your query will be passed on to these search engines.)

Let us know what you think! Comments and suggestions are welcome.

US006324534B1

(12) **United States Patent**
Neal et al.

(10) **Patent No.: US 6,324,534 B1**
(45) **Date of Patent: Nov. 27, 2001**

(54) **SEQUENTIAL SUBSET CATALOG SEARCH ENGINE**

(75) Inventors: **Michael Renn Neal**, Superior; **James Michael Wilmsen**, Westminster; **Christopher Wade Beall**, Lafayette, all of CO (US)

(73) Assignee: **Requisite Technology, Inc.**, Westminster, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: ~~09/393,994~~

(22) Filed: **Sep. 10, 1999**

(51) **Int. Cl.**[7] **G06F 17/30**

(52) **U.S. Cl.** **707/3**; 707/1; 707/102; 707/103; 706/12; 706/45; 709/218; 709/224; 709/245

(58) **Field of Search** 345/419, 450, 345/850, 853, 854; 707/1, 2, 3, 4, 5, 6, 10, 100, 101, 102, 205, 513; 709/224, 218, 245; 706/12, 45

(56) [illegible]

ENTS

4,468, ... 707/1
5,630, ... 707/103
5,706, ... 707/5
5,781, et al. 707/3
5,806, ... 707/3
5,832, ... 707/205
5,924, ... 707/5

age.)

NS

Kao, Davi earch in Multivari-
ate Data", ternational Confer-
ence on Sci e Management, Jul.
1–3, 1998,

Lee, Jinho et al., "Integrating Structured Data and Text: A Multi–dimensional Approach", Proceedings of the 2000 International Conference on Information Technology: Coding and Computing, Mar. 27–29, 2000, pp. 264–269.*

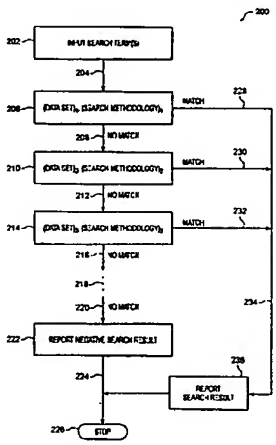
Park, Sanghyun et al., "Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases", Proceedings of the 16th International Conference on Data Engineering, Feb. 29 –Mar. 3, 2000, pp. 23–32.*

Primary Examiner—Hosain T. Alam
Assistant Examiner—Shahid Alam
(74) *Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman LLP

(57) **ABSTRACT**

An electronic catalog search engine is configurable to optimize the search process by identifying the desired item from the most advantageous supplier, while efficiently utilizing computing resources. The search engine comprises a configurable search and data subset creation mechanism. The system accepts search terms from a user, and then executes a sequence of search strategies on subsets of the database which may include a proximity search, a word count search, and a fuzzy logic search. Subsets can be searched in any order and different search strategies can be applied to different subsets. The sequences are terminated when search steps have uncovered at least one match. Each database entry has a corresponding product category. A list of categories from each of the matching products is dynamically compiled and displayed to the user. The user can page through the list of displayed matches, or alternatively can create a subset of the list by selecting only the items within one of the categories. In addition, the user can further refine the list of items by selecting those items having a particular attribute. The invention has the advantage that users with a wide range of skills and/or familiarity with products can quickly find the products that they need. The system has the additional feature of creating electronic requisitions for the products listed in the database.

13 Claims, 3 Drawing Sheets



US006321224B1

(12) **United States Patent**
Beall et al.

(10) **Patent No.: US 6,321,224 B1**
(45) **Date of Patent: Nov. 20, 2001**

(54) **DATABASE SEARCH, RETRIEVAL, AND CLASSIFICATION WITH SEQUENTIALLY APPLIED SEARCH ALGORITHMS**

(75) Inventors: **Christopher Wade Beall**, Lafayette; **Michael Renn Neal**, Superior; **James Michael Wilmsen**, Westminster, all of CO (US)

(73) Assignee: **Requisite Technology, Inc.**, Westminster, CO (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/514,524**

(22) Filed: **Feb. 28, 2000**

Related U.S. Application Data

(63) Continuation of application No. 09/058,553, filed on Apr. 10, 1998, now Pat. No. 6,032,145.

(51) **Int. Cl.**[7] **G06F 17/30**
(52) **U.S. Cl.** **707/5**; 707/3; 707/6
(58) **Field of Search** 707/5, 3, 6

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,879,648		11/1989	Cochran et al.	364/300
4,947,028		8/1990	Gorog	235/381
4,984,155	*	1/1991	Geier et al.	364/401
4,992,940		2/1991	Dworkin	364/401
5,206,949		4/1993	Cochran et al.	395/600
5,231,566		7/1993	Blutinger et al.	364/401
5,319,542		6/1994	King, Jr. et al.	364/401
5,630,125		5/1997	Zellweger	395/614
5,715,444	*	2/1998	Danish et al.	395/604
5,799,157		8/1998	Escallon	395/227
5,924,090	*	7/1999	Krellenstein	707/5
5,995,979	*	11/1999	Cochran	707/104
6,032,145	*	2/2000	Beall et al.	707/5
6,169,992		1/2001	Beall et al.	707/103
6,230,154	*	5/2001	Raz et al.	707/3

FOREIGN PATENT DOCUMENTS

WO 99/53421 10/1999 (WO) G06F/17/30

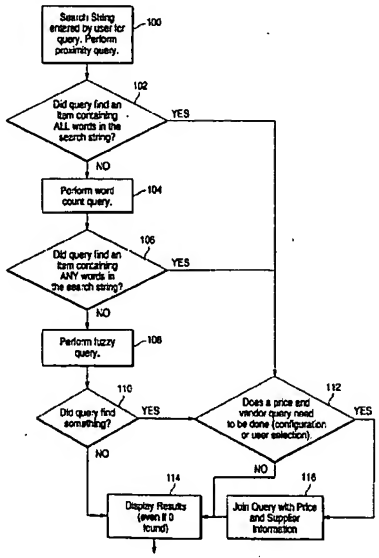
* cited by examiner

Primary Examiner—Paul R. Lintz
(74) *Attorney, Agent, or Firm*—Blakely, Sokoloff, Taylor & Zafman LLP

(57) **ABSTRACT**

An electronic catalog requisition system includes software for efficiently selecting items from a database. The software accepts search terms from a user, and then executes a sequence of search strategies on the database which may include a proximity search, a word count search, and a fuzzy logic search. The sequence is terminated when a search algorithm has uncovered at least one match. Each database entry has a corresponding product category. A list of categories from each of the matching products is dynamically compiled and displayed to the user. The user can page through the list of displayed matches, or alternatively can create a subset of the list by selecting only the items within one of the categories. In addition, the user can further refine the list of items by selecting those items having a particular attribute. The software can also maintain a list of synonyms for attributes as an aid for finding appropriate matches within the database. Natural adjectives to certain of the attributes are recognized as further refinements to the search criteria. The invention has the advantage that users with a wide range of skills and/or familiarity with products can quickly find the products that they need. The software has the additional feature of creating electronic requisitions for the products listed in the database.

132 Claims, 6 Drawing Sheets



US006032145A

United States Patent [19]

Beall et al.

[11] Patent Number: 6,032,145

[45] Date of Patent: Feb. 29, 2000

[54] METHOD AND SYSTEM FOR DATABASE MANIPULATION

[75] Inventors: Christopher Wade Beall, Lafayette; Michael Renn Neal, Superior; James Michael Wilmsen, Westminster, all of Colo.

[73] Assignee: Requisite Technology, Inc., Westminster, Colo.

[21] Appl. No.: 09/058,553

[22] Filed: Apr. 10, 1998

[51] Int. Cl.[7] G06F 17/30

[52] U.S. Cl. 707/5; 707/3; 707/104; 705/26; 705/27

[58] Field of Search 707/3, 104, 103, 707/5; 705/26, 27

[56] References Cited

U.S. PATENT DOCUMENTS

4,879,648	11/1989	Cochran et al. .	
4,947,028	8/1990	Gorog	235/280
4,984,155	1/1991	Geier et al. .	
4,992,940	2/1991	Dworkin	705/26
5,206,949	4/1993	Cochran et al. .	
5,231,566	7/1993	Blutinger et al. .	
5,319,542	6/1994	King, Jr. et al.	705/27
5,630,125	5/1997	Zellweger	707/103
5,715,444	2/1998	Danish et al. .	
5,799,157	8/1998	Escallon	705/27
5,897,622	4/1999	Blinn et al.	705/26

OTHER PUBLICATIONS

Lintz, Search Page US 5, 581, 757, Mar. 5, 1996, pp. 1–2.

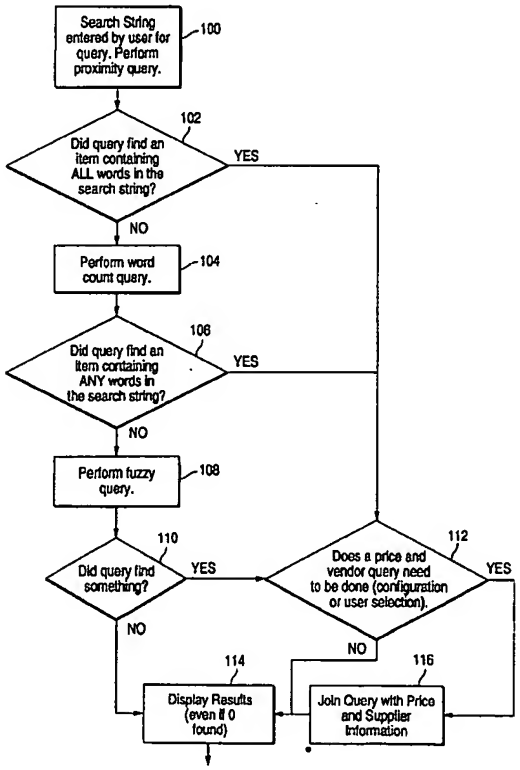
Primary Examiner—Paul R. Lintz

Attorney, Agent, or Firm—Cooley Godward LLP

[57] ABSTRACT

An electronic catalog requisition system includes software for efficiently selecting items from a database. The software accepts search terms from a user, and then executes a sequence of search strategies on the database which may include a proximity search, a word count search, and a fuzzy logic search. The sequence is terminated when a search algorithm has uncovered at least one match. Each database entry has a corresponding product category. A list of categories from each of the matching products is dynamically compiled and displayed to the user. The user can page through the list of displayed matches, or alternatively can create a subset of the list by selecting only the items within one of the categories. In addition, the user can further refine the list of items by selecting those items having a particular attribute. The software can also maintain a list of synonyms for attributes as an aid for finding appropriate matches within the database. Natural adjectives to certain of the attributes are recognized as further refinements to the search criteria. The invention has the advantage that users with a wide range of skills and/or familiarity with products can quickly find the products that they need. The software has the additional feature of creating electronic requisitions for the products listed in the database.

44 Claims, 6 Drawing Sheets



Title: *Rating the Metasearch Engines* , By: Jacsó, Péter, Information Today, 8755-6286, December 1, 2001, Vol. 18, Issue 11
Database: *Academic Search Premier*

Section: Internet Today

Internet Insights

Rating the Metasearch Engines

Contents

It's time to pay some well-deserved attention to these helpful resources

Amid the flurry of new search engines that are making waves-such as Teoma and WiseNut, which are striving to topple Google from its top position--the metasearch engines (with the possible exception of Vivísimo) seem to get less attention. Many of the once-celebrated search engines have had hardly more than 15 minutes of fame. Does anyone remember Oingo or Simpli, the wunder-engines of last year? Maybe metasearch engines don't get that much attention because of their secondary nature. After all, they mostly ride on the coattails of the primary search engines.

Universal Metasearch Engines

I liked using metasearch engines even before Nature published Steve Lawrence and C. Lee Giles' 1999 article "Accessibility of Information on the Web." The researchers' findings concluded that even the largest search engines alone don't cover more than one-sixth of the possible Web sites. Librarians probably weren't really surprised by the news, as they know that analogous traditional resources--such as abstracting-and-indexing databases--rarely cover more than a quarter of the printed literature of their specialty area. That's even more true for databases that cover as wide a spectrum as the Web.

If you want to do thorough research you had better use not only ABI/INFORM but three or four other business databases, such as H.W. Wilson's Business Abstracts, Gale Group's Trade & Industry database, and Responsive Database Services' Business & Industry file to get decently comprehensive coverage.

Considering the ever-expanding Webscape, using a single search engine can't yield comprehensive results. True, probably no one would miss results from the Irrelevant Web (à la Invisible Web), which forms a very large proportion of the entire Web universe, but relying on a single search engine would leave many valuable sites uncovered by the searcher.

Universal metasearch engines have helped to fill the gaps by running queries against several of the largest search engines that claim universal coverage. Those could be the most useful metasearch engines, which would include the best universal search engines: Northern Light, Google, AltaVista, and

FAST, to name a few. The fact is, however, that even if some metasearch engines claim they cover Google, chances are they don't, or won't be able to for long, since Google is known to put out the software equivalent of a "No Trespassing" sign for crawlers--and it enforces the principle. Northern Light isn't keen on having crawlers visit its database for harvesting, either. The exceptions are those cases in which the metasearch engine developer has an agreement with Google or Northern Light.

Disappearing Acts and Features

In preparing for a workshop at the Internet Librarian conference, I looked up many of the metasearch engines that I've used in the past. I was surprised to see how many of them went south in just the past few months. For example, Justseek is a good one, but it has recently given me the "File 403 Not Authorized" message. Meta IQ had a good mix of search engines and directories with good search-refinement options, but it became an irritating shopping bot. Metafetcher, a site that claims to be "the biggest on the Web" and covers more than 300 search engines, has been working sporadically for weeks. E-mails to the Webmaster remain unanswered. Too bad, since it has an option for personalizing the list of engines to be searched, and saves the choices for the next visit. The screen comes up, but the customization option doesn't work. Mamma, "the mother of all search engines," floods you with so many paid listings that it makes midnight infomercials seem informative and substantial. Dogpile, which brandishes the tag line "all results, no mess," is also guilty of an excessive amount of paid listings. At least it offers decent customization, which allows the user to choose which sites' findings will appear on the first, second, etc., page of the results list, or not at all.

Meta-Metasearch Engine

I expected that someone would start harvesting the metasearch engines, and indeed CleverSearch has done just that. It searches, or claims to search, the biggest metasearch engines, such as MetaCrawler, Mamma, Dogpile, and Metafind. It also claims to cover Google, but when you limit the search to Google using the query term "money" it returns no results. (See Figures 1 and 2.) There's no search engine that wouldn't have a few million sites that match this search term. Actually, when searching Google directly, the engine retrieves over 31 million pages. I don't know of other such meta-metasearch engines, but technically it's not a big deal. Instead of having a series of simple JavaScript or PHP codes to skim the primary search engines, a meta-metasearch engine would search a set of databases built by metasearch engines.

The Google Boast

It's hard to resist including the Google name in a metasearch engine's repertoire, as it's without a doubt the most popular search engine. Ithaki, a relative newcomer, for example, happily lists Google among the engines it searches. When you click on the Google name in the results list (see Figure 3), you'll be taken to Yahoo!'s home page. This is nonsense to start with. All other metasearch engines that offer such a link take you to the record in the primary search engine. True, Yahoo! uses Google (by agreement) as a search engine to complement its directory, but that doesn't mean that Ithaki indeed accesses Google under Yahoo!.

To illustrate my point, I searched for a term ("supercalifragile," the Italian version of Mary Poppins' "supercalifragilisticexpialidocious") that I hoped wouldn't flood me with results, to see what Ithaki's response would be. Well, it didn't retrieve a single hit from Google, which, in a direct search, found 190 hits. The veteran MetaCrawler search engine also claims to cover Google (see Figure 4), but the searches--limited to Google in the power search mode--for "Lanikai," a splendid beach area in Hawaii (see Figure 5), and "supercalifragile" bring back no results from Google, which has 10,100 hits for the former term and 190 for the latter in its native search mode.

Vivace, Molto Vivace, Vivissimo

There are good metasearch engines, however. My two favorites are CNET's Search.com (known before its acquisition as Savvy Search), and Profusion, which was acquired by IntelliSeek. Both of them offer

excellent search-customization features. They represent the first-generation metasearch engines, but I find them far more efficient than the dozens of Young Turks in the metasearch league.

But Vivísimo is, indeed, an outstanding contender. In my test searches it lived up to its recently acquired fame. I'm pleased that the company had the smarts to reserve the URL with the much better-known spelling of the word, "vivissimo." Classical music fans (which I don't claim to be) know this word, which means that the identified part is to be performed in a very lively fashion.

Vivísimo does a very lively job and has a fresh, good design. However, it's not as revolutionary as the reviews contend. It allows you, for example, to take a sneak peek at the sites that appear in the results list by opening a small window from within the list without leaving it. This is very useful, but not a brand-new feature. The little-known Surfy search engine (which sounds like Snow White's eighth dwarf) has been offering this feature, and so has WiseNut.

In Vivísimo, the clustering into topics of results on the side of the screen also contributes to its efficiency. Once again, results clustering is not new--Northern Light has been doing that for many years by collocating results on the fly into folders. It even got a patent for it. Vivísimo certainly uses a different algorithm to generate the cluster names and to reduce the risk of being challenged by patent disputes. (See Figure 6.) The choice of terms extracted from the source is not as good as the terms in Northern Light. On the other hand, the summary detailing the contribution of the individual search engines to the result is excellent and highly informative.

There's an even higher-IQ metasearch engine that doesn't get much attention from the press. Its name is SurfWax, and it has remarkable features for previewing and summarizing results, as well as for saving pages that the user designates. It's a tad intimidating in its complexity, though. To use some of its advanced features you must register, and for some other features you need to pay a modest yearly fee, which is a limiting factor.

All in all, Vivísimo is perhaps the most powerful and most appealing metasearch engine. I visit it often. I didn't abandon Search.com and Profusion because they offer vertical metasearching, a feature that's getting more and more important, and one that Vivísimo doesn't have yet. Vertical metasearching allows users to search several sites that remain invisible to the crawlers of most search engines by querying such sites in one fell swoop as the best encyclopedias, directories, and other ready-reference sources that are every librarian's bread and butter. They're worth discussing in a separate column.

PHOTO (BLACK & WHITE): Figure 1

PHOTO (BLACK & WHITE): Figure 2

PHOTO (BLACK & WHITE): Figure 3

PHOTO (BLACK & WHITE): Figure 4

PHOTO (BLACK & WHITE): Figure 5

PHOTO (BLACK & WHITE): Figure 6

~~~~~~~~

By Péter Jacsó

Péter Jacsó is associate professor of library and information science at the University of Hawaii's Department of Information and Computer Sciences. His email address is jacso@hawaii.edu
~~~~~~~~

Source: Information Today, Dec2001, Vol. 18 Issue 11, p28, 2p, 6bw.
Item Number: 5640401

Top of Page

Formats: HTML Full Text PDF Full Text Citation
Result List | Refine Search Print E-mail Save

Title: *HOW A SEARCH ENGINE WORKS* , By: Liddy, Elizabeth, Searcher, 1070-4795, May 1, 2001, Vol. 9, Issue 5
Database: *Academic Search Premier*

HOW A SEARCH ENGINE WORKS

Contents

Search engine is the popular term for an information retrieval (IR) system. While researchers and developers take a broader view of IR systems, consumers think of them more in terms of what they want the systems to do -- namely search the Web, or an intranet, or a database. Actually consumers would really prefer a finding engine, rather than a search engine.

Search engines match queries against an index that they create. The index consists of the words in each document, plus pointers to their locations within the documents. This is called an inverted file. A search engine or IR system comprises four essential modules:

- A document processor
- A query processor
- A search and matching function
- A ranking capability

While users focus on "search," the search and matching function is only one of the four modules. Each of these four modules may cause the expected or unexpected results that consumers get when they use a search engine.

Document Processor

The document processor prepares, processes, and inputs the documents, pages, or sites that users search against. The document processor performs some or all of the following steps:

- Normalizes the document stream to a predefined format.
- Breaks the document stream into desired retrievable units.
- Isolates and metatags subdocument pieces.
- Identifies potential indexable elements in documents.
- Deletes stop words.
- Stems terms.
- Extracts index entries.
- Computes weights.
- Creates and updates the main inverted file against which the search engine searches in order to match queries to documents.

Steps 1-3: Preprocessing. While essential and potentially important in affecting the outcome of a search, these first three steps simply standardize the multiple formats encountered when deriving documents from various providers or handling various Web sites. The steps serve to merge all the data into a single consistent data structure that all the downstream processes can handle. The need for a well-formed, consistent format is of relative importance in direct proportion to the sophistication of later steps of document processing. Step two is important because the pointers stored in the inverted file will enable a system to retrieve various sized units -- either site, page, document, section, paragraph, or sentence.

Step 4: Identify elements to index. Identifying potential indexable elements in documents dramatically affects the nature and quality of the document representation that the engine will search against. In designing the system, we must define the word "term." Is it the alpha-numeric characters between blank spaces or punctuation? If so, what about non-compositional phrases (phrases in which the separate words do not convey the meaning of the phrase, like "skunk works" or "hot dog"), multi-word proper names, or inter-word symbols such as hyphens or apostrophes that can denote the difference between "small business men" versus small-business men." Each search engine depends on a set of rules that its document processor must execute to determine what action is to be taken by the "tokenizer," i.e. the software used to define a term suitable for indexing.

Step 5: Deleting stop words. This step helps save system resources by eliminating from further processing, as well as potential matching, those terms that have little value in finding useful documents in response to a customer's query. This step used to matter much more than it does now when memory has become so much cheaper and systems so much faster, but since stop words may comprise up to 40 percent of text words in a document, it still has some significance. A stop word list typically consists of those word classes known to convey little substantive meaning, such as articles (a, the), conjunctions (and, but), interjections (oh, but), prepositions (in, over), pronouns (he, it), and forms of the "to be" verb (is, are). To delete stop words, an algorithm compares index term candidates in the documents against a stop word list and eliminates certain terms from inclusion in the index for searching.

Step 6: Term Stemming. Stemming removes word suffixes, perhaps recursively in layer after layer of processing. The process has two goals. In terms of efficiency, stemming reduces the number of unique words in the index, which in turn reduces the storage space required for the index and speeds up the search process. In terms of effectiveness, stemming improves recall by reducing all forms of the word to a base or stemmed form. For example, if a user asks for analyze, they may also want documents which contain analysis, analyzing, analyzer, analyzes, and analyzed. Therefore, the document processor stems document terms to analy- so that documents which include various forms of analy- will have equal likelihood of being retrieved; this would not occur if the engine only indexed variant forms separately and required the user to enter all. Of course, stemming does have a downside. It may negatively affect precision in that all forms of a stem will match, when, in fact, a successful query for the user would have come from matching only the word form actually used in the query.

Systems may implement either a strong stemming algorithm or a weak stemming algorithm. A strong stemming algorithm will strip off both inflectional suffixes (-s, -es, -ed) and derivational suffixes (-able, -aciousness, -ability), while a weak stemming algorithm will strip off only the inflectional suffixes (-s, -es, -ed).

Step 7: Extract index entries. Having completed steps 1 through 6, the document processor extracts the remaining entries from the original document. For example, the following paragraph shows the full text sent to a search engine for processing:

Milosevic's comments, carried by the official news agency Tanjug, cast doubt over the governments at the talks, which the international community has called to try to prevent an all-out war in the Serbian province. "President Milosevic said it was well known that Serbia and Yugoslavia were firmly committed to resolving problems in Kosovo, which is an integral part of Serbia, peacefully in Serbia with the participation of the representatives of all ethnic communities," Tanjug said. Milosevic was speaking during a meeting with British Foreign Secretary Robin Cook, who delivered an ultimatum to attend negotiations in a week's time on an autonomy proposal for Kosovo with ethnic Albanian leaders from the province. Cook earlier told a conference that Milosevic had agreed to study the proposal.

Steps 1 to 6 reduce this text for searching to the following: Milosevic comm carri offic new agen Tanjug cast doubt govern talk interna commun call try prevent all-out war Serb province President Milosevic said well known Serbia Yugoslavia firm commit resolv problem Kosovo integr part Serbia peace Serbia particip representa ethnic commun Tanjug said Milosevic speak meeti British Foreign Secretary Robin Cook deliver ultimat attend negoti week time autonomy propos Kosovo ethnic Alban lead province Cook earl told conference Milosevic agree study propos.

The output of step 7 is then inserted and stored in an inverted file that lists the index entries and an indication of their position and frequency of occurrence. The specific nature of the index entries, however, will vary based on the decision in Step 4 concerning what constitutes an "indexable term." More sophisticated document processors will have phrase recognizers, as well as Named Entity recognizers and Categorizers, to insure index entries such as Milosevic are tagged as a Person and entries such as Yugoslavia and Serbia as Countries.

Step 8: Term weight assignment. Weights are assigned to terms in the index file. The simplest of search engines just assign a binary weight: 1 for presence and 0 for absence. The more sophisticated the search engine, the more complex the weighting scheme. Measuring the frequency of occurrence of a term in the document creates more sophisticated weighting, with length-normalization of frequencies still more sophisticated. Extensive experience in information retrieval research over many years has clearly demonstrated that the optimal weighting comes from use of "tf/idf." This algorithm measures the frequency of occurrence of each term within a document. Then it compares that frequency against the frequency of occurrence in the entire database.

Not all terms are good "discriminators" -- that is, all terms do not single out one document from another very well. A simple example would be the word "the." This word appears in too many documents to help distinguish one from another. A less obvious example would be the word "antibiotic." In a sports database when we compare each document to the database as a whole, the term "antibiotic" would probably be a good discriminator among documents, and therefore would be assigned a high weight. Conversely, in a database devoted to health or medicine, "antibiotic" would probably be a poor discriminator, since it occurs very often. The TF/IDF weighting scheme assigns higher weights to those terms that really distinguish one document from the others.

Step 9: Create index. The index or inverted file is the internal data structure that stores the index information and that will be searched for each query. Inverted files range from a simple listing of every alpha-numeric sequence in a set of documents/pages being indexed along with the overall identifying numbers of the documents in which the sequence occurs, to a more linguistically complex list of entries, the tf/idf weights, and pointers to where inside each document the term occurs. The more complete the information in the index, the better the search results.

Query Processor

Query processing has seven possible steps, though a system can cut these steps short and proceed to match the query to the inverted file at any of a number of places during the processing. Document processing shares many steps with query processing. More steps and more documents make the process more expensive for processing in terms of computational resources and responsiveness. However, the longer the wait for results, the higher the quality of results. Thus, search system designers must choose what is most important to their users -- time or quality. Publicly available search engines usually choose time over very high quality, having too many documents to search against.

The steps in query processing are as follows (with the option to stop processing and start matching indicated as "Matcher"):

- Tokenize query terms.
- Recognize query terms vs. special operators.

--- Matcher

- Delete stop words.
- Stem words.
- Create query representation.

— Matcher

- Expand query terms.
- Compute weights.

— Matcher

Step 1: Tokenizing. As soon as a user inputs a query, the search engine -- whether a keyword-based system or a full natural language processing (NLP) system -- must tokenize the query stream, i.e., break it down into understandable segments. Usually a token is defined as an alpha-numeric string that occurs between white space and/or punctuation.

Step 2: Parsing. Since users may employ special operators in their query, including Boolean, adjacency, or proximity operators, the system needs to parse the query first into query terms and operators. These operators may occur in the form of reserved punctuation (e.g., quotation marks) or reserved terms in specialized format (e.g., AND, OR). In the case of an NLP system, the query processor will recognize the operators implicitly in the language used no matter how the operators might be expressed (e.g., prepositions, conjunctions, ordering).

At this point, a search engine may take the list of query terms and search them against the inverted file. In fact, this is the point at which the majority of publicly available search engines perform the search.

Steps 3 and 4: Stop list and stemming. Some search engines will go further and stop-list and stem the query, similar to the processes described above in the Document Processor section. The stop list might also contain words from commonly occurring querying phrases, such as, "I'd like information about." However, since most publicly available search engines encourage very short queries, as evidenced in the size of query window provided, the engines may drop these two steps.

Step 5: Creating the query. How each particular search engine creates a query representation depends on how the system does its matching. If a statistically based matcher is used, then the query must match the statistical representations of the documents in the system. Good statistical queries should contain many synonyms and other terms in order to create a full representation. If a Boolean matcher is utilized, then the system must create logical sets of the terms connected by AND, OR, or NOT.

An NLP system will recognize single terms, phrases, and Named Entities. If it uses any Boolean logic, it will also recognize the logical operators from Step 2 and create a representation containing logical sets of the terms to be AND'd, OR'd, or NOT'd.

At this point, a search engine may take the query representation and perform the search against the inverted file. More advanced search engines may take two further steps.

Step 6: Query expansion. Since users of search engines usually include only a single statement of their information needs in a query, it becomes highly probable that the information they need may be expressed using synonyms, rather than the exact query terms, in the documents which the search engine searches against. Therefore, more sophisticated systems may expand the query into all possible synonymous terms and perhaps even broader and narrower terms.

This process approaches what search intermediaries did for end users in the earlier days of commercial search systems. Back then, intermediaries might have used the same controlled vocabulary or thesaurus used by the indexers who assigned subject descriptors to documents. Today, resources such as WordNet are generally available, or specialized expansion facilities may take the initial query and enlarge it by

adding associated vocabulary.

Step 7: Query term weighting (assuming more than one query term). The final step in query processing involves computing weights for the terms in the query. Sometimes the user controls this step by indicating either how much to weight each term or simply which term or concept in the query matters most and must appear in each retrieved document to ensure relevance.

Leaving the weighting up to the user is not common, because research has shown that users are not particularly good at determining the relative importance of terms in their queries. They can't make this determination for several reasons. First, they don't know what else exists in the database, and document terms are weighted by being compared to the database as a whole. Second, most users seek information about an unfamiliar subject, so they may not know the correct terminology.

Few search engines implement system-based query weighting, but some do an implicit weighting by treating the first term(s) in a query as having higher significance. The engines use this information to provide a list of documents/pages to the user.

After this final step, the expanded, weighted query is searched against the inverted file of documents.

Search and Matching Function

How systems carry out their search and matching functions differs according to which theoretical model of information retrieval underlies the system's design philosophy. Since making the distinctions between these models goes far beyond the goals of this article, we will only make some broad generalizations in the following description of the search and matching function. Those interested in further detail should turn to R. Baeza-Yates and B. Ribeiro-Neto's excellent textbook on IR (Modern Information Retrieval, Addison-Wesley, 1999).

Searching the inverted file for documents meeting the query requirements, referred to simply as "matching," is typically a standard binary search, no matter whether the search ends after the first two, five, or all seven steps of query processing. While the computational processing required for simple, unweighted, non-Boolean query matching is far simpler than when the model is an NLP-based query within a weighted, Boolean model, it also follows that the simpler the document representation, the query representation, and the matching algorithm, the less relevant the results, except for very simple queries, such as one-word, non-ambiguous queries seeking the most generally known information.

Having determined which subset of documents or pages matches the query requirements to some degree, a similarity score is computed between the query and each document/page based on the scoring algorithm used by the system. Scoring algorithms rankings are based on the presence/absence of query term(s), term frequency, tf/idf, Boolean logic fulfillment, or query term weights. Some search engines use scoring algorithms not based on document contents, but rather, on relations among documents or past retrieval history of documents/pages.

After computing the similarity of each document in the subset of documents, the system presents an ordered list to the user. The sophistication of the ordering of the documents again depends on the model the system uses, as well as the richness of the document and query weighting mechanisms. For example, search engines that only require the presence of any alpha-numeric string from the query occurring anywhere, in any order, in a document would produce a very different ranking than one by a search engine that performed linguistically correct phrasing for both document and query representation and that utilized the proven tf/idf weighting scheme.

However the search engine determines rank, the ranked results list goes to the user, who can then simply click and follow the system's internal pointers to the selected document/page.

More sophisticated systems will go even further at this stage and allow the user to provide some relevance feedback or to modify their query based on the results they have seen. If either of these are

available, the system will then adjust its query representation to reflect this value-added feedback and re-run the search with the improved query to produce either a new set of documents or a simple re-ranking of documents from the initial search.

What Document Features Make a Good Match to a Query

We have discussed how search engines work, but what features of a query make for good matches? Let's look at the key features and consider some pros and cons of their utility in helping to retrieve a good representation of documents/pages.

• **Term frequency:** How frequently a query term appears in a document is one of the most obvious ways of determining a document's relevance to a query. While most often true, several situations can undermine this premise. First, many words have multiple meanings -- they are polysemous. Think of words like "pool" or "fire." Many of the non-relevant documents presented to users result from matching the right word, but with the wrong meaning.

Also, in a collection of documents in a particular domain, such as education, common query terms such as "education" or "teaching" are so common and occur so frequently that an engine's ability to distinguish the relevant from the non-relevant in a collection declines sharply. Search engines that don't use a tf/idf weighting algorithm do not appropriately down-weight the overly frequent terms, nor are higher weights assigned to appropriate distinguishing (and less frequently-occurring) terms, e.g., "early-childhood."

- **Location of terms:** Many search engines give preference to words found in the title or lead paragraph or in the metadata of a document. Some studies show that the location -- in which a term occurs in a document or on a page -- indicates its significance to the document. Terms occurring in the title of a document or page that match a query term are therefore frequently weighted more heavily than terms occurring in the body of the document. Similarly, query terms occurring in section headings or the first paragraph of a document may be more likely to be relevant.
- **Link analysis:** Web-based search engines have introduced one dramatically different feature for weighting and ranking pages. Link analysis works somewhat like bibliographic citation practices, such as those used by Science Citation Index. Link analysis is based on how well-connected each page is, as defined by Hubs and Authorities, where Hub documents link to large numbers of other pages (out-links), and Authority documents are those referred to by many other pages, or have a high number of "in-links" (J. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms. 1998, pp. 668-77).
- **Popularity:** Google and several other search engines add popularity to link analysis to help determine the relevance or value of pages. Popularity utilizes data on the frequency with which a page is chosen by all users as a means of predicting relevance. While popularity is a good indicator at times, it assumes that the underlying information need remains the same.
- **Date of Publication:** Some search engines assume that the more recent the information is, the more likely that it will be useful or relevant to the user. The engines therefore present results beginning with the most recent to the less current.
- **Length:** While length per se does not necessarily predict relevance, it is a factor when used to compute the relative merit of similar pages. So, in a choice between two documents both containing the same query terms, the document

that contains a proportionately higher occurrence of the term relative to the length of the document is assumed more likely to be relevant.

- **Proximity of query terms:** When the terms in a query occur near to each other within a document, it is more likely that the document is relevant to the query than if the terms occur at greater distance. While some search engines do not recognize phrases per se in queries, some search engines clearly rank documents in results higher if the query terms occur adjacent to one another or in closer proximity, as compared to documents in which the terms occur at a distance.
- **Proper nouns** sometimes have higher weights, since so many searches are performed on people, places, or things. While this may be useful, if the search engine assumes that you are searching for a name instead of the same word as a normal everyday term, then the search results may be peculiarly skewed. Imagine getting information on "Madonna," the rock star, when you were looking for pictures of madonnas for an art history class.

Summary

The above explanation lays out the range of processing that might occur in a search engine, along with the many options that a search engine provider decides on. The range of options may help clarify users' frequent surprise at the results their queries return. Up till now, search engine providers have mainly opted for less, versus more, complex processing of documents and queries. The typical search results therefore leave a lot of work to be done by the searcher, who must wend their way through the results, clicking on and exploring a number of documents before finding exactly what they seek. The typical evolution of products and services suggests that this status-quo will not continue. Search engines that go further in the complexity and quality of the processing performed will be rewarded with greater allegiance by searchers, as well as financially rewarding opportunities to serve as the search engine on more organizations' intranets.

Searchers should keep watching for the best and pursuing it.

~~~~~~~~

By Elizabeth Liddy, Director of the Center for Natural Language Processing Professor, School of Information Studies Syracuse University

Source: Searcher, May2001, Vol. 9 Issue 5, p38, 6p.
Item Number: 4446940

Top of Page

Formats: HTML Full Text Citation
Result List | Refine Search Print E-mail Save
~~~~~~~~

SUMMARY PERFORMANCE COMPARISONS TREC-2 THROUGH TREC-8

Karen Sparck Jones
Computer Laboratory, University of Cambridge

December 8, 1999

The context

This comparison series has attempted to illustrate long-term TREC trends, as embodied in the results for the baseline Adhoc task. As in last year's comparisons, covering TREC-2 - TREC-7, from TREC-5 onwards there has been a more careful separation of different *versions* of the topics, ranging from Very short (titles only) to Long (titles, descriptions and narratives), and between automatic and manual *modes* of query formulation: see the detail given in Table 1.

While last year's comparisons (Appendix B, TREC-7 Proceedings) gave performance details for the whole series from TREC-2 onwards, this year's detail is restricted to TREC-7 and TREC-8 only. First, the way the TREC-6 topics were formed could lead to titles and descriptions that were viewed as complementary rather than as less or more inclusive: this meant that controlled study of the effects of increasing topic length and detail was impossible. In TREC-7 and TREC-8 title terms are included in descriptions (so the difference between descriptions and titles+descriptions is in term frequency for the queries): TREC-7 and TREC-8 therefore supply two cycles of testing on the same topic basis. At the same time, it is evident from the detailed results for these two cycles in Table 2 that there is little difference in performance, whether of best levels or (to a considerable extent) by hardy perennial teams. The TREC-8 results can therefore be seen as a 'wind-up' on the long programme of Adhoc evaluations with the 'traditional' TREC data, and the end of a phase that is also signalled by the fact that evaluation with this type of data is being mothballed for TREC-9.

Table entries

Table 2 follows the same conventions as in previous summaries. Thus the detailed figures are taken from the Working Notes, and cover only the better performing, not all, the teams.

The conventions are as follows: figures are not rounded; performance is assigned to 'blocks'; teams per block are NOT in merit order, but in in Working Notes results order; where there is more than one run per team the best is taken, regardless of the particular strategy used. Simple, hopefully sufficiently identifiable, short names have been given to the teams (with some streamlining where teams have changed name or composition over the years).

TABLE 1 : TOPIC DETAILS

Topic fields available as base for queries, TREC-2 - TREC-7 :

	(TREC-1	TREC-2	TREC-3	TREC-4	TREC-5	TREC-6	TREC-7	TREC-8
T= title	x	x	x		x	x	x	x
D= description	x	x	x	x	x	x	x	x
N= narrative	x	x	x		x	x	x	x
C= concepts	x	x						

Average topic and field length :

Total	107.4	130.8	103.4	16.3	82.7	88.4	57.6	51.8
T	3.8	4.9	6.5	-	3.8	2.7	2.5	2.5
D	17.9	18.7	22.3	16.3	15.7	20.4	14.3	13.8
N	64.5	78.8	74.6	-	63.2	65.3	40.8	35.5
C	21.2	28.5	-	-	-	-	-	-

TABLE 2 : RETRIEVAL PERFORMANCE, TREC-7, TREC-8

TREC ADHOC SEARCH RESULTS FOR PRECISION AT DOCUMENT CUTOFF 30

KEY TO TABLE NOTATIONS :

a = fully automatic searches
m = manual searches

V	= very short queries, i.e.	title only from topics, aka	T
S	= short queries	description only	D
M	= medium queries	title+description	T+D
L	= long queries	title+description+narrative	T+D+N

/contd

	TREC-7 a V	TREC-7 a S	TREC-7 a M	TREC-7 a L	TREC-7 m L	TREC-8 a V	TREC-8 a S	TREC-8 a M	TREC-8 a L	TREC-8 m L
>=60										ManInst
>=55					Clarit					IITetc
>=50					ManInst Waterlo					Oracle
>=45					GMUetc					Clarit GEetc
>=40		NEC	ATT Cityetc UMass	BBN Cityetc NEC UMass	ANU Harris Berkely Toronto	CUNY		ATT FUB IBMTJWs Msoft MIT CUNY	FUB Fujitsu Msoft MIT CUNY Neuchat	
>=35	Cityetc	Cornell CUNY Fujitsu	Lexis RMIT	ANU Cornell CUNY IRIT TwentyO Iowa	GEetc Lexis	ATT Fujitsu IBMTJWs Msoft MultTxt RICOH Sab/Crn	UMass	Fujitsu GEetc IBMTJWg IRIT MultTxt NTT Sab/Crn TwentyO Neuchat UMass Twente	ACSys GEetc IRIT JHopk NTT Sab/Crn UMass	
>=30	ATT Cornell CUNY Fujitsu Lexis NEC NTTData RMIT Waterlo	IBMTJWs	IBMTJWg IRIT	GMUetc NTTData Rutgers Berkely UNC	FS	ACSys RMIT TwentyO UMass	IBMTJWs Sab/Crn	ACSys CMU IITetc ImperC JHopk RICOH RMIT Marylnd	RMIT	
>=25	ANU Avignon GEetc IBMTJWg ETH Berkely Marylnd			FUB ImperC JHopk NSA		City/M		UNCy	CMU Dartmth	

Performance summary

To give a final overview of performance from TREC-2 - TREC-8, Table 3 gives the highest level of performance reached in each TREC for the various versions and modes.

As this table clearly shows, the early TRECs with 'good' topics reached high levels of performance in both automatic and manual modes; performance in the middle TRECs declined under the much less favourable data conditions (whether of topic information or relevant document accessibility); then in TREC-7 and TREC-8 performance for automatic mode in particular revived. This must be attributed to superior systems, since best manual performance has remained on a plateau. More specifically, amplifying on Tables 2 and 3, it is clear that the better level of performance in the TREC-7 and TREC-8 evaluations was the same.

TABLE 3 : PERFORMANCE SUMMARY

Highest level reached, Precision at Document Cutoff 30, TREC-2 - TREC-8

	V T a	S D a	M T+D a	L T+D+N a	L T+D+N m
>= 65					
>= 60				333	333 888
>= 55					222 777
>= 50				222	666
>= 45					444 555
>= 40	888	777 888	444 777 888	777 888	
>= 35	777				
>= 30	666			555 666	
>= 25		555 666			
>= 20					

Key: 222 = TREC-2 highest performance level, 333 = TREC-3 ditto, etc

(TREC-2 included Concept field
TREC-4 manual did not have Narrative field)

Overall comments

As before, but even more clearly when the evidence of TREC-8 is added in,

1. Many teams obtain similar performance, even at top levels.
2. Manual query formation can give superior performance to automatic, typically reflecting the amount of effort put in and/or user judgements on intermediate outputs.
3. There has been some convergence, especially in automatic searching, on default strategies; but similar performance is also obtained with very different strategies, presumably reflecting the dominating influence of the frequency data that strategies share.
4. Results in TREC generally illustrate the way in which established teams can maintain and enhance their performance; but it also shows that new teams can take advantage of published TREC experience and the rich training data that is available to get up to speed quickly.
5. Performance is broadly correlated with the quality of the topic information available and the difficulty of the topics.
6. However, as the results for TREC-7 and TREC-8 show, it is possible to do almost as well in automatic searching with the minimal (the Very short title) topics as with much longer ones.
7. The best levels of automatic search performance as illustrated by TREC-7 and TREC-8 are quite respectable, and in particular in many cases are achieved with relatively simple, albeit well-motivated, methods. It may be noted that at Cutoff 10, several teams achieved almost 50% Precision in automatic searching even with the Very short titles in TREC-8, and several reached more than 50% with the Medium length titles+descriptions. Manual searching without enormous effort can do better, achieving 70%, but the time and attention required is nevertheless not negligible.

```
Set     Items   Description
S1     126810   (SEARCH? OR QUERY?)(2N)(TOOL? ? OR ENGINE? OR SOFTWARE? OR
           PAGE? OR SYSTEM? OR APPLICATION?) OR METASEARCH? OR (IR OR IN-
           FORMATION()RETRIEVAL)(2N)(TOOL? OR MODULE? OR APPLICATION? OR
           PROGRAM? OR SOFTWARE? OR SYSTEM? OR SITE? OR WEBSITE?)
S2     290715   (MULTIPLE OR MULTIPLICITY OR PLURAL? OR MANY OR SEVERAL OR
           DIFFERENT OR VARIOUS OR VARIETY)(3N)(STEP OR MODULE? OR SEQUE-
           NC? OR STEPS OR PROCESSES OR PROCEDURES?)
S3    6334184   SEQUENTIAL OR NEXT OR STEPWISE? OR STEPS OR SEQUENCE? OR D-
           IMENSION? OR MULTIDIMENSION? OR CLASS OR CLASSES
S4        469   S1 AND S2 AND S3
S5         25   S4 AND (FUZZY OR BOOLEAN OR STEMMING OR WILDCARD OR TRUNCA-
           T? OR PLURALS OR FIELD(2N)(LIMIT? OR DELIMIT?) OR LINGUISTIC?
           OR MORPHOLOG? OR MORPHEME? OR THESAURUS OR SYNONYM? OR THESAU-
           RI)
S6        123   S4 AND (TEXT? OR DOCUMENT? OR PAGE? OR WEBPAGE? OR PARAGRA-
           PH? OR TITLE? OR ABSTRACT? OR PUBLICATION?)
S7       2553   S1 AND FUZZY
S8        111   S7 AND (SYNONYM? OR THESAURUS OR THESAURI? OR SIMILAR()TER-
           M? ?)
S9          0   S8 AND (WILDCARD? OR STEM? OR TRUNCAT? OR WILD()CARD? ?)
S10         0   S8 AND (MISPELL? OR ALTERNATIVE()SPELLING OR BRITISH()SPEL-
           LING OR PLURAL? OR VARIATION?)
S11        21   S7 AND (MISPELL? OR ALTERNATIVE()SPELLING OR BRITISH()SPEL-
           LING OR PLURAL? OR VARIATION?)
S12        46   S5 OR S11
S13        41   RD (unique items)
S14        35   S13 NOT PY>2001
S15      6027   S1 AND (INTELLIGENT? OR SMART? OR MULTIFACET? OR METASEARC-
           H? OR MULTI()STEP? OR MULTISTEP?)
S16      1259   S1 AND ITERAT?
S17         9   S6 AND (S16 OR S15)
S18      3199   (S16 OR S15) AND (TEXT? OR DOCUMENT? OR PAGE? OR PARAGRAPH?
            OR PUBLICATION? OR CITATION? OR ABSTRACT? OR FULLTEXT?)
S19      1736   S18 AND (INTERNET? OR INTRANET? OR WWW OR WORLDWIDEWEB OR -
           WEB OR ONLINE OR ON()LINE? OR SERVER? OR NETWORK?)
S20       226   (S2 OR S3) AND S19
S21        57   S20 AND (REPEAT? OR REITERAT? OR NEXT? OR PROGRESSION OR S-
           EQUENTIAL)
S22        50   RD (unique items)
S23        40   S22 NOT PY>2001
S24        38   S23 NOT (S17 OR S14)
File   8:Ei Compendex(R) 1970-2005/May W5
         (c) 2005 Elsevier Eng.  Info. Inc.
File  35:Dissertation Abs Online 1861-2005/May
         (c) 2005 ProQuest Info&Learning
File  65:Inside Conferences 1993-2005/Jun W1
         (c) 2005 BLDSC all rts. reserv.
File   2:INSPEC 1969-2005/May W5
         (c) 2005 Institution of Electrical Engineers
File  94:JICST-EPlus 1985-2005/Apr W3
         (c)2005 Japan Science and Tech Corp(JST)
File 111:TGG Natl.Newspaper Index(SM) 1979-2005/Jun 09
         (c) 2005 The Gale Group
File   6:NTIS 1964-2005/May W5
         (c) 2005 NTIS, Intl Cpyrght All Rights Res
File 144:Pascal 1973-2005/May W5
         (c) 2005 INIST/CNRS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
File  34:SciSearch(R) Cited Ref Sci 1990-2005/Jun W1
         (c) 2005 Inst for Sci Info
File  99:Wilson Appl. Sci & Tech Abs 1983-2005/May
         (c) 2005 The HW Wilson Co.
File  95:TEME-Technology & Management 1989-2005/May W1
```

24/5/24 (Item 4 from file: 2)
DIALOG(R)File 2:INSPEC

6582037 INSPEC Abstract Number: C2000-06-7250R-014
Title: Retrieval engines **and the** search engine **food chain**
Author(s): Arnold, S.E.
Conference Title: 20th Annual National Online Meeting. Proceeedings-1999 p.7-15
Editor(s): Williams, M.E.
Publisher: Information Today, Medford, NJ, USA
Publication Date: 1999 Country of Publication: USA xi+548 pp.
Material Identity Number: XX-1999-02806
Conference Title: Proceedings of 20th National Online Meeting
Conference Sponsor: Information Today; Lexis-Nexis
Conference Date: 18-20 May 1999 Conference Location: New York, NY, USA
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: **Online** search-and-retrieval has not been a part of mainstream computing. Search-and-retrieval, as well as advanced search-and-retrieval functions, are becoming ubiquitous. The rarefied system services like group collaboration, **document** management, and natural language processing are finding their way into new operating systems. Consequently, innovation will move to hitherto advanced search-and-retrieval services in hopes of crafting the **next** Yahoo. Visualization, **intelligent** agents, filtering, and hybrid systems will provide a wealth of functionality to **online** users. The proven **tools** of **search** -and-retrieval, namely, asking someone for information, will not fall into disuse. The user will have a richer set of tools to use from a standard Windows desktop or from within the browser interface. We are entering a new era of opportunity for information management and navigation. (0 Refs)
Subfile: C
Descriptors: information retrieval; **Internet** ; **search engines**
Identifiers: retrieval engines; **search engine** ; information retrieval; group collaboration; **document** management; natural language processing; operating systems; **intelligent** agents; information filtering; hybrid systems; Windows; **Web** browser; **Internet**
Class Codes: C7250R (Information retrieval techniques); C7250N (Search engines); C7210N (Information networks)

24/5/25 (Item 5 from file: 2)
DIALOG(R)File 2:INSPEC

6412027 INSPEC Abstract Number: C2000-01-7250N-001
Title: "DISKo Iskatel" - easy metasearch **on the** Internet
Author(s): Bogdanov, V.M.
Author Affiliation: Comput. Press Journal, Moscow, Russia
Journal: Komp'yuter Press no.8 p.152-3
Publisher: Komp'yut. Press,
Publication Date: Aug. 1999 Country of Publication: Russia
CODEN: KOPRFZ ISSN: 0868-6157
SICI: 0868-6157(199908)8L.152:TIEM;1-5
Material Identity Number: G475-1999-009
Language: Russian Document Type: Journal Paper (JP)
Treatment: Practical (P); Product Review (R)
Abstract: "DISKo Iskatel" (DISCo Finder) is a new program from the DISCo Company. This **tool searches** for information on several **servers** . The main difference between DISKo Iskatel and other software of this **class** is the possibility of storing search parameters and results and reusing them in the **next** run. You can reorganize the results of the search as you like. Especially useful is the possibility of "hiding" **servers** or **pages** which are of no interest. As a result, you will see only those links that you need. During the search, you can request a validity check of the links obtained from the **search engines** . This allows you to see which links lead to wrong or missing **pages** without calling them up on your browser. By double-clicking any **page** node, the default **Internet** browser is invoked. There is also the possibility of reorganizing bookmarks in **Internet** Explorer by importing them into DISKo Iskatel, checking their validity, reorganizing them, and exporting them back to **Internet** Explorer. (0 Refs)
Subfile: C
Descriptors: hypermedia; **Internet** ; microcomputer applications; **online** front-ends; **search engines** ; software reviews
Identifiers: DISKo Iskatel; **Internet metasearching** ; DISCo Finder; search parameters; search results storage; search results reorganization; **Web server** hiding; **Web page** hiding; hyperlinks; World Wide **Web** ; validity check; missing **pages** ; default **Internet** browser; bookmark reorganization; **Internet** Explorer
Class Codes: C7250N (Search engines); C0310H (Equipment and software evaluation methods); C7210N (Information networks)

24/5/32 (Item 3 from file: 144)
DIALOG(R)File 144:Pascal

15005832 PASCAL No.: 01-0161606
The Web in 2010 : Challenges and opportunities for database research Informatics 10 years back, 10 years ahead : Dagstuhl, 27-31 August 2000
WEIKUM Gerhard
WILHELM Reinhard, ed
University of the Saarland, Saarbruecken, Germany
International conference and Research Center for Computer Science (Schloss Dagstuhl DEU) 2000-08-27
Journal: Lecture notes in computer science, 2001, 2000 1-23
ISBN: 3-540-41635-8 ISSN: 0302-9743 Availability: INIST-16343; 354000092042400010
No. of Refs.: 52 ref.
Document Type: P (Serial); C (Conference Proceedings) ; A (Analytic)
Country of Publication: Germany
Language: English
The impressive advances in global **networking** and information technology provide great opportunities for all kinds of **Web** -based information services, ranging from digital libraries and information discovery to virtual-enterprise workflows and electronic commerce. However, many of these services still exhibit rather poor quality in terms of unacceptable performance during load peaks, frequent and long outages, and unsatisfactory search results. For the **next** decade, the overriding goal of database research should be to provide means for building zero-administration, self-tuning information services with predictable response time, virtually continuous availability, and, ultimately, "money-back" service-quality guarantees. A particularly challenging aspect of this theme is the quality of search results in digital libraries, scientific data repositories, and on the **Web** . To aim for more **intelligent** search that can truly find needles in haystacks, classical information retrieval methods should be integrated with querying capabilities for structurally richer **Web** data, most notably XML data, and automatic classification methods based on standardized ontologies and statistical machine learning. This paper gives an overview of promising research directions along these lines.

English Descriptors: Distance measurement; Groupware; Electronic **document** management **system** ; Workflow; **Information retrieval** ; Performance evaluation; Database; Information technology; Selftuning control; Time response; Information flow; Peak load; Tuning; Response time; Information service; Availability; Service quality; **Internet** ; World wide **web** ; Digital information

French Descriptors: Mesure de distance; Collecticiel; Systeme gestion electronique **document** ; Workflow; Recherche information; Evaluation performance; Base donnee; Technologie information; Commande autoajustable ; Reponse temporelle; Flux information; Regime pointe; Accord frequence; Temps reponse; Service information; Disponibilite; Qualite service; **Internet** ; Reseau **WWW** ; Information numerique

Classification Codes: 001D02B07D; 001D04B03

```
Set     Items   Description
S1     126810   (SEARCH? OR QUERY?)(2N)(TOOL? ? OR ENGINE? OR SOFTWARE? OR
           PAGE? OR SYSTEM? OR APPLICATION?) OR METASEARCH? OR (IR OR IN-
           FORMATION()RETRIEVAL)(2N)(TOOL? OR MODULE? OR APPLICATION? OR
           PROGRAM? OR SOFTWARE? OR SYSTEM? OR SITE? OR WEBSITE?)
S2     290715   (MULTIPLE OR MULTIPLICITY OR PLURAL? OR MANY OR SEVERAL OR
           DIFFERENT OR VARIOUS OR VARIETY)(3N)(STEP OR MODULE? OR SEQUE-
           NC? OR STEPS OR PROCESSES OR PROCEDURES?)
S3    6334184   SEQUENTIAL OR NEXT OR STEPWISE? OR STEPS OR SEQUENCE? OR D-
           IMENSION? OR MULTIDIMENSION? OR CLASS OR CLASSES
S4        469   S1 AND S2 AND S3
S5         25   S4 AND (FUZZY OR BOOLEAN OR STEMMING OR WILDCARD OR TRUNCA-
           T? OR PLURALS OR FIELD(2N)(LIMIT? OR DELIMIT?) OR LINGUISTIC?
           OR MORPHOLOG? OR MORPHEME? OR THESAURUS OR SYNONYM? OR THESAU-
           RI)
S6        123   S4 AND (TEXT? OR DOCUMENT? OR PAGE? OR WEBPAGE? OR PARAGRA-
           PH? OR TITLE? OR ABSTRACT? OR PUBLICATION?)
S7       2553   S1 AND FUZZY
S8        111   S7 AND (SYNONYM? OR THESAURUS OR THESAURI? OR SIMILAR()TER-
           M? ?)
S9          0   S8 AND (WILDCARD? OR STEM? OR TRUNCAT? OR WILD()CARD? ?)
S10         0   S8 AND (MISPELL? OR ALTERNATIVE()SPELLING OR BRITISH()SPEL-
           LING OR PLURAL? OR VARIATION?)
S11        21   S7 AND (MISPELL? OR ALTERNATIVE()SPELLING OR BRITISH()SPEL-
           LING OR PLURAL? OR VARIATION?)
S12        46   S5 OR S11
S13        41   RD (unique items)
S14        35   S13 NOT PY>2001
File   8:Ei Compendex(R) 1970-2005/May W5
         (c) 2005 Elsevier Eng.  Info. Inc.
File  35:Dissertation Abs Online 1861-2005/May
         (c) 2005 ProQuest Info&Learning
File  65:Inside Conferences 1993-2005/Jun W1
         (c) 2005 BLDSC all rts. reserv.
File   2:INSPEC 1969-2005/May W5
         (c) 2005 Institution of Electrical Engineers
File  94:JICST-EPlus 1985-2005/Apr W3
         (c)2005 Japan Science and Tech Corp(JST)
File 111:TGG Natl.Newspaper Index(SM) 1979-2005/Jun 09
         (c) 2005 The Gale Group
File   6:NTIS 1964-2005/May W5
         (c) 2005 NTIS, Intl Cpyrght All Rights Res
File 144:Pascal 1973-2005/May W5
         (c) 2005 INIST/CNRS
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 1998 Inst for Sci Info
File  34:SciSearch(R) Cited Ref Sci 1990-2005/Jun W1
         (c) 2005 Inst for Sci Info
File  99:Wilson Appl. Sci & Tech Abs 1983-2005/May
         (c) 2005 The HW Wilson Co.
File  95:TEME-Technology & Management 1989-2005/May W1
         (c) 2005 FIZ TECHNIK
```

14/5/1 (Item 1 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

06317339 E.I. No: EIP03107389284
Title: A family of similarity and star products: New additions to fuzzy **relational products**
Author: Santiprabhob, Pratit
Corporate Source: Intelligent Systems Laboratory Faculty of Science and Technology Assumption University, Bangkok, Thailand
Conference Title: Joint 9th IFSA World Congress and 20th NAFIPS International Conference
Conference Location: Vancouver, BC, Canada Conference Date: 20010725-20010728
Sponsor: International Fuzzy Systems Association; North American Fuzzy Information Processing Society
E.I. Conference No.: 60785
Source: Annual Conference of the North American Fuzzy Information Processing Society - NAFIPS v 4 2001. p 2013-2018 (IEEE cat n 01TH8569)
Publication Year: 2001
Language: English
Document Type: CA; (Conference Article) Treatment: T; (Theoretical)
Journal Announcement: 0303W3
Abstract: In this paper, a family of new **fuzzy** relational products is presented. This new family of products is based on the Similarity product and the Star product previously defined by the author and his colleagues. The products have been developed from practical viewpoints arisen in certain applications for which the semantics of the classical **fuzzy** relational products has proven somewhat inadequate. These two products are further extended here. The definitions and semantics of the two products together with those of their respective **variations** are elaborated. Suggestions on the products' potential application are also discussed. These new additions together with the classical **fuzzy** relational products would certainly contribute greatly to our ability to manipulate and process information with uncertainty by means of **fuzzy** relations. 15 Refs.
Descriptors: ***Fuzz** y sets; Data processing; **Information retrieval** ; Semantics; Uncertain **systems**
Identifiers: **Fuzzy** relational products; Similarity product; Star product; **Fuzzy** relation; Qualitative information processing
Classification Codes:
921.6 (Numerical Methods); 723.2 (Data Processing); 903.3 (Information Retrieval & Use); 903.2 (Information Dissemination)
921 (Applied Mathematics); 723 (Computer Software, Data Handling & Applications); 903 (Information Science)
92 (ENGINEERING MATHEMATICS); 72 (COMPUTERS & DATA PROCESSING); 90 (ENGINEERING, GENERAL)

14/5/12 (Item 12 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

00823797 E.I. Monthly No: EI7906043566 E.I. Yearly No: EI79045269
Title: Interrogation of Data Banks in Natural Language in a Complex Cognitive System.
Title: L'INTERROGATION DES BANQUES DE DONNEES EN LANGAGE NATUREL DANS UNE DEMARCHE COGNITIVE COMPLEXE.
Author: Guenoche, A.; Virbel, J.
Corporate Source: CNRS, Marseille, Fr
Source: Information Processing & Management v 15 n 1 1979 p 33-46
Publication Year: 1979
CODEN: IPMADK ISSN: 0306-4573
Language: FRENCH
Journal Announcement: 7906
Abstract: This paper deals with **linguistic** and mathematical problems which arise in natural language processing of data bases of information, liable to have a cognitive function. Within the framework of the project AVEROES (French acronym: Experimental Analysis and Validation of Reasoning in Work in the Social Sciences), questions are asked of a data base providing a description of **morphological** aspects of roman amphora, in order to analyze and to verify traditional typologies in this archaeological field. To begin with, a language for the fomulation of questions is defined; it is planned to associate a question in natural language with the elements concerned by it, that is, the objects dealt with the " support " corresponding to a data array copied out from the bank) and the **different procedures** to be used (the " theme " corresponding to a set of algorithms). **Next** , it is shown that the matching of algorithms with data arrays, first comes to a problem of choice between representations, between algorithms, and then is likely to lead to uninteresting answers, or even false ones if questions convey an implicit knowledge that is not or cannot be put into words. 14 refs. In French with English abstract.
Descriptors: ***INFORMATIO N RETRIEVAL SYSTEMS**
Identifiers: NATURAL LANGUAGE PROCESSING
Classification Codes:
901 (Engineering Profession); 723 (Computer Software)
90 (GENERAL ENGINEERING); 72 (COMPUTERS & DATA PROCESSING)

14/5/18 (Item 4 from file: 2)
DIALOG(R)File 2:INSPEC

5471222 INSPEC Abstract Number: C9702-7250R-011
Title: Fuzzy **full-text searches in OCR databases**
Author(s): Myka, A.; Guntzer, U.
Author Affiliation: Wilhelm-Schickard-Inst. fur Inf., Tubingen Univ., Germany
Conference Title: Digital Libraries. Research and Technology Advances. ADL '95 Forum. Selected Papers p.131-45
Editor(s): Adam, N.R.; Bhargava, B.K.; Halem, M.; Yesha, Y.
Publisher: Springer-Verlag, Berlin, Germany
Publication Date: 1996 Country of Publication: Germany xiii+290 pp.
ISBN: 3 540 61410 9 Material Identity Number: XX95-01098
Conference Title: Proceedings of a Forum on Research and Technology Advances in Digital Libraries
Conference Sponsor: NASA
Conference Date: 15-17 May 1995 Conference Location: McLean, VA, USA
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P); Experimental (X)
Abstract: We describe several methods of **fuzzy** full-text searching that can be used in OCR databases. The preliminary results of our tests show that it is possible to increase recall to a certain extent without losing too much precision. However, because almost every increase of recall has to be paid for by a decrease of precision, the user or administrator, respectively, of an **information retrieval system** has to decide individually on the importance of each of these values: if the risk of decreasing precision should be minimized, a slight increase of recall can be achieved by means of using character classes. If the highest possible recall (together with a reasonable precision) is necessary, employment of a confusion table may be the appropriate choice. Of course, in the case when significant loss of precision is introduced into the system by means of a fault-tolerant string matching algorithm, additional tools have to be provided to ensure that a user is not flooded with unnecessary information, e.g., a KWIC index could be used in order to provide for a preliminary overview of results. The tests are still extended at the moment. Extensions include the size of the analyzed full-text database as well as the addition of **variations** of some of the described methods, especially those based on linear scanning. However, such a test can never be complete: different algorithms can be evaluated as well as **variations** and combinations of the ones described here. Furthermore, OCR errors are, to a certain extent, dependent on the OCR device. Therefore, different results could possibly be achieved by means of testing data from different OCR devices. (0 Refs)
Subfile: C
Descriptors: character sets; full-text databases; **fuzzy** logic; optical character recognition; query formulation; string matching
Identifiers: **fuzzy** full-text searches; OCR databases; recall; precision ; **information retrieval system** ; character classes; confusion table; fault-tolerant string matching algorithm; KWIC index; linear scanning
Class Codes: C7250R (Information retrieval techniques)

14/5/19 (Item 5 from file: 2)
DIALOG(R)File 2:INSPEC

5422097 INSPEC Abstract Number: C9612-7250L-008
Title: Witco's technical information system
Author(s): Eckard, A.; Costello, M.T.; Thalman, W.J.; Weaver, J.A.
Journal: Inform vol.10, no.8 p.32, 34-6
Publisher: Assoc. Inf. & Image Manag. Int,
Publication Date: Sept. 1996 Country of Publication: USA
CODEN: INFREN ISSN: 0892-3876
SICI: 0892-3876(199609)10:8L.32:WTIS;1-J
Material Identity Number: K884-96002
Language: English Document Type: Journal Paper (JP)
Treatment: Applications (A)
Abstract: The conversion of disorganized paper documents into a highly structured, easily searchable electronic database not only fosters easy access of information through the use of document management software, but also archives the documents into a compact, easily protected format. Using the suite of ZyImage software applications, and a minimum of personnel resources, we were able to build an R&D database by converting the paper files into an electronic format and saving the documents to appropriate LAN media and locations. By incorporating the newer electronic files into the database, we can immediately have access to all current R&D activities and information. ZyBuild (the indexing software) automatically builds or updates the index to include every word it finds in the document it is indexing. This process saves many man-hours that would otherwise be necessary to manually key-word each document. This also allows a full-text search to be performed. ZyFindi (the **search software**) has many **search** operators that can be utilized for finding information. We can search on a word or phrase, use Boolean operators, proximity, quorum, numeric or date range or key fields. We can also perform searches with the **fuzzy** search option; this allows a level of ambiguity to compensate for **variations** in spelling or OCR errors. When paper documents are converted into an electronic format suitable for searching, the conversion isn't completely accurate; this is not acceptable for experimental data. The ability to link the original scanned image to the searchable intelligent document assures complete retention of the original data. (0 Refs)
Subfile: C
Descriptors: data conversion; document image processing; full-text databases; indexing; petroleum industry; visual databases
Identifiers: Witco; technical information system; paper documents; searchable electronic database; document management software; document archiving; easily protected format; ZyImage software applications; personnel resources; R&D database; file conversion; document saving; LAN media; ZyBuild indexing software; full-text search; ZyFindi **search software** ; search operators; **fuzzy** search option; conversion accuracy; scanned image; searchable intelligent document; data retention
Class Codes: C7250L (Non-bibliographic retrieval systems); C5260B (Computer vision and image processing techniques); C6130D (Document processing techniques); C6160S (Spatial and pictorial databases); C7160 (Manufacturing and industrial administration)

14/5/21 (Item 7 from file: 2)
DIALOG(R)File 2:INSPEC

02221196 INSPEC Abstract Number: C84018661
Title: On-line catalogues: responding to the challenge
Author(s): Horny, K.L.
Journal: Canadian Library Journal vol.40, no.5 p.277-82
Publication Date: Oct. 1983 Country of Publication: Canada
CODEN: CLIJBX ISSN: 0008-4352
Language: English Document Type: Journal Paper (JP)
Treatment: General, Review (G); Practical (P)
Abstract: The rapid growth of the on-line catalogue has led to many **variations** , some more 'user friendly' than others. The author describes the features of some systems and approaches used to guide the public through the maze of 'Boolean operators', ' **fuzzy** matches' and 'menus.'. (3 Refs)
Subfile: C
Descriptors: cataloguing; information retrieval; **information retrieval systems**
Identifiers: online public access catalogues; on-line catalogue; Boolean operators; **fuzzy** matches; menus
Class Codes: C7210 (Information services and centres); C7240 (Information analysis and indexing); C7250 (Information storage and retrieval)

14/5/25 (Item 3 from file: 94)
DIALOG(R)File 94:JICST-EPlus

00759507 JICST ACCESSION NUMBER: 89A0504642 FILE SEGMENT: JICST-E
Significance of user aspects in information retrieval systems .
HOSONO KIMIO (1)
(1) Keio Univ., Faculty of Literature
Joho Kanri(Journal of Information Processing and Management), 1989, VOL.32,NO.6, PAGE.489-500, FIG.3, TBL.1, REF.14
JOURNAL NUMBER: F0392AAX ISSN NO: 0021-7298
UNIVERSAL DECIMAL CLASSIFICATION: 002.5:005
LANGUAGE: Japanese COUNTRY OF PUBLICATION: Japan
DOCUMENT TYPE: Journal
ARTICLE TYPE: Original paper
MEDIA TYPE: Printed Publication
ABSTRACT: This paper describes, first of all, some basic problems of typical **information retrieval systems** in terms of, 1) retrieval techniques based on the binary logic and exact match approaches, and 2) search formulation procedures based on the assumption that users are able to specify their information needs clearly and completely, which in fact is not necessarily likely. Secondly, it mentions several retrieval techniques, such as a cosine measure between a document and query, **fuzzy** and probabilistic retrieval, and the one constructed from a neural network, and then, a means to represent queries in graph, which are all expected to give some breakthroughs for developing promising methods to solve those problems. Finally, it reviews fundamental functions required for a highly user friendly interface of a **information retrieval system** and an expert system for providing referral information into which some of those functions are incorporated.(author abst.)
DESCRIPTORS: **information retrieval system** ; Boolean algebra; matching; information need; user; modeling; information retrieval; diversification; subject; data retrieval; keyword; matching(graph); membership function; expert system
BROADER DESCRIPTORS: information system; computer application system; system; lattice(mathematics); algebraic system; demand; operation(processing); retrieval; **variation** ; fact retrieval; vocabulary; function(mathematics); mapping(mathematics); artificial intelligence system
CLASSIFICATION CODE(S): AC06020S

14/5/33 (Item 5 from file: 34)
DIALOG(R)File 34:SciSearch(R) Cited Ref Sci

05516421 Genuine Article#: WD675 Number of References: 49
Title: Comparing Boolean and probabilistic information retrieval systems **across queries and disciplines**
Author(s): Losee RM (REPRINT)
Corporate Source: UNIV N CAROLINA,MANNING HALL, CB 3360/CHAPEL HILL//NC/27599 (REPRINT)
Journal: JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, 1997, V48 , N2 (FEB), P143-156
ISSN: 0002-8231 Publication date: 19970200
Publisher: JOHN WILEY & SONS INC, 605 THIRD AVE, NEW YORK, NY 10158-0012
Language: English Document Type: ARTICLE
Geographic Location: USA
Subfile: CC SOCS--Current Contents, Social & Behavioral Sciences;
Journal Subject Category: COMPUTER SCIENCE, INFORMATION SYSTEMS
Abstract: Whether using Boolean queries or ranking documents using document and termweights will result in better retrieval performance has been the subject of considerable discussion among document retrieval system users and researchers. We suggest a method that allows one to analytically compare the two approaches to retrieval and examine their relative merits. The performance of **information retrieval systems** may be determined either by using experimental simulation, or through the application of analytic techniques that directly estimate the retrieval performance, given values for query and database characteristics. Using these performance predicting techniques, sample performance figures are provided for queries using the Boolean and and or, aswell as for probabilistic systems assuming statistical term independence or term dependence. The **variation** of performance across sublanguages (used in different academic disciplines) and queries is examined. The performance of models failing to meet statistical and other assumptions is examined.
Identifiers--KeyWord Plus(R): RELEVANCE FEEDBACK; DOCUMENT-RETRIEVAL; TERM DEPENDENCE; SEARCH TERMS; **FUZZY** SET; PERFORMANCE; ALGORITHM; RANKING; MODELS
Research Fronts: 95-0540 001 (**SOFTWARE** REUSE; PROBABILISTIC **INFORMATION - RETRIEVAL** ; TOPICAL RELEVANCE RELATIONSHIPS; MEDLINE SEARCHING; MODELING COORDINATION; OBJECT-ORIENTED TECHNOLOGY)
Cited References:
BARTELL BT, 1994, P173, ACM ANN C RES DEV IN
BECHER T, 1987, V12, P261, STUD HIGH EDUC
BELKIN NJ, 1993, P339, ACM ANN C RES DEV IN
BONZI S, 1984, V40, P247, J DOC
BOOKSTEIN A, 1983, V34, P331, J AM SOC INFORM SCI
BOOKSTEIN A, 1985, P11, P 8 ANN INT ACM SIGI
BOOKSTEIN A, 1982, RES DEV INFORMATION
BOVEY JD, 1984, V3, P84, INFORM TECHNOL R & D
CHOW CK, 1968, V14, P462, IEEE T INFORMATION T
CHOW D, 1982, V29, P127, J ASSOC COMPUT MACH
COOPER WS, 1995, V13, P100, ACM T INFORM SYST
CROFT WB, 1986, V37, P71, J AM SOC INFORM SCI
EVANS R, 1994, P121, P 15 NAT ONL M MEDF
EVERETT DM, 1992, V43, P658, J AM SOC INFORM SCI
FOX EA, 1990, V41, P10, J AM SOC INFORM SCI
GUPTA PD, 1987, V38, P245, J AM SOC INFORM SCI
HAAS SW, 1995, P137, P 4 ANN S DOC ANAL I
KNEALE WC, 1986, DEV LOGIC
LAM K, 1982, V7, P500, ACM T DATABASE SYST
LEE JH, 1994, P182, ACM ANN C RES DEV IN
LEE JH, 1995, P180, ACM ANN C RES DEV IN
LOSEE RM, 1986, P258, ACM C RES DEV INFORM
LOSEE RM, 1991, V27, P1, INFORM PROCESS MANAG
LOSEE RM, 1994, V30, P193, INFORM PROCESS MANAG

LOSEE RM, 1994, V30, P293, INFORM PROCESS MANAG
LOSEE RM, 1995, V31, P555, INFORM PROCESS MANAG
LOSEE RM, 1988, V24, P315, INFORMATION PROCESSI
LOSEE RM, 1988, V39, P8, J AM SOC INFORM SCI
LOSEE RM, 1995, V46, P519, J AM SOC INFORM SCI
LOSEE RM, 1996, V47, P95, J AM SOC INFORM SCI
LOSEE RM, 1995, P265, P 5 INT C SCI INT C
MARKEY K, 1981, ONTAP ONLINE TRAININ
MOON SB, 1993, THESIS U N CAROLINA
NIE JY, 1989, V25, P477, INFORM PROCESS MANAG
PORTER MF, 1980, V14, P130, PROGRAM
RADECKI T, 1979, V15, P247, INFORMATION PROCESSI
RADECKI T, 1982, V33, P365, J AM SOC INFORM SCI
ROBERTSON SE, 1990, P153, INFORMATICS 10 PROSP
ROBERTSON SE, 1976, V27, P129, J AM SOC INFORM SCI
ROBERTSON SE, 1977, V33, P294, J DOC
ROBERTSON SE, 1986, V12, P71, J INFORM SCI
SAGER N, 1981, P199, P 44 ASIS ANN M WHIT
SALTON G, 1984, 84588 TR CORN U
SMEATON AF, 1984, V3, P15, INFORM TECHNOL R & D
SPINK A, 1995, V31, P161, INFORM PROCESS MANAG
TIBBO HR, 1992, V14, P31, LIBR INFORM SCI RES
TURTLE H, 1994, P212, 17TH P ANN INT ACM S
VANRIJSBERGEN CJ, 1977, V33, P106, J DOC
YU CT, 1983, V2, P129, INFORM TECHNOL R & D

```
Set     Items   Description
S1      16249   (SEARCH? OR QUERY?)(2N)(TOOL? ? OR ENGINE? OR SOFTWARE? OR
             PAGE? OR SYSTEM? OR APPLICATION?) OR METASEARCH? OR (IR OR IN-
             FORMATION()RETRIEVAL)(2N)(TOOL? OR MODULE? OR APPLICATION? OR
             PROGRAM? OR SOFTWARE? OR SYSTEM? OR SITE? OR WEBSITE?)
S2      44426   (MULTIPLE OR MULTIPLICITY OR PLURAL? OR MANY OR SEVERAL OR
             DIFFERENT OR VARIOUS OR VARIETY)(3N)(STEP OR SEQUENC? OR STEPS
              OR PROCESSES OR PROCEDURES?)
S3    1095679   SEQUENTIAL OR NEXT OR STEPS OR DIMENSION? OR MULTIDIMENSIO-
             N? OR CLASS OR CLASSES
S4         56   S1 AND S2 AND S3
S5         39   S4 AND IC=(G06F OR H04L)
S6          3   S4 AND IC=G06F-007
S7          2   S6 NOT AD>20011217
S8          0   S5 AND (FUZZY OR BOOLEAN OR STEMMING OR PLURALS OR TRUNCAT-
             ION OR FIELD(2N)(LIMIT? OR DELIMIT?) OR LINGUISTIC? OR MORPHO-
             LOG?)
S9         16   S5 AND (INTERNET? OR INTRANET? OR WWW OR WEB OR WORLDWIDEW-
             EB OR WAN OR LAN OR NETWORK?)
S10        15   S9 AND IC=G06F
S11         7   S10 AND (TEXT? OR DOCUMENT? OR OBJECT? OR RETRIEV? OR RESU-
             LT?)
S12         7   S11 NOT S7
S13         4   S12 NOT AD=20011217:20041217
S14         4   S13 NOT AD=20041217:20050701
S15       293   S1 AND (FUZZY OR BOOLEAN OR STEMMING OR PLURALS OR TRUNCAT-
             ION OR FIELD(2N)(LIMIT? OR DELIMIT?) OR SYNONYM? OR THESAURUS
             OR LINGUISTIC? OR MORPHOLOG?)
S16       267   S15 AND IC=G06F
S17       199   S16 AND (INTERNET? OR TEXT? OR DOCUMENT? OR WWW OR WORLDWI-
             DEWEB? OR WEB OR WAN OR LAN OR NETWORK? OR ONLINE? OR ON()LIN-
             E? OR REMOTE?)
S18         1   S17 AND S2
S19        13   S17 AND S3
S20        13   S18 OR S19
S21        12   S20 NOT (S14 OR S7)
S22        35   S16 AND IC=G06F-007
S23        34   S22 NOT S20
S24        34   S23 NOT (S14 OR S7)
S25        18   S24 NOT AD>20011217
S26    112477   STEM OR STEMS OR STEMMING OR STEMMED OR TRUNCAT? OR WILDCA-
             RD? OR WILD()CARD?
S27      8834   FUZZY
S28      1562   SYNONYM? OR THESAURUS? OR THESAURI
S29         0   S26 AND S27 AND S28
S30        16   S26 AND S28
S31         0   S30 AND S1
File 347:JAPIO Nov 1976-2005/Feb(Updated 050606)
         (c) 2005 JPO & JAPIO
File 350:Derwent WPIX 1963-2005/UD,UM &UP=200536
         (c) 2005  Thomson Derwent
```

21/5/3 (Item 3 from file: 347)
DIALOG(R)File 347:JAPIO

06673845 **Image available**
INFORMATION GENERATION **SYSTEM** , **INFORMATION RETRIEVAL SYSTEM** AND RECORDING MEDIUM

PUB. NO.: 2000-259671 [JP 2000259671 A]
PUBLISHED: September 22, 2000 (20000922)
INVENTOR(s): FUJIOKA TAKAKO
APPLICANT(s): DAINIPPON PRINTING CO LTD
APPL. NO.: 11-067318 [JP 9967318]
FILED: March 12, 1999 (19990312)
INTL CLASS: **G06F-017/30**

ABSTRACT

PROBLEM TO BE SOLVED: To provide an information generation system capable of indexing a **document** so as to easily identify information included in the **document** and to provide an **information retrieval system** to which a user's viewpoint in retrieving work can be inputted and which has high efficiency of retrieving work.

SOLUTION: A computer presents a keyword database including keywords included in an information **document** to be provided (step 601). A user selects and inputs a keyword in which the user is interested (step 602). The computer presents related work and **synonym** information corresponding to the inputted keyword (step 603). The user refers to the presented related work information and inputs a keyword of the user's interest (**steps** 605, 606). The computer displays the contents of a **document** file to which the inputted related word is indexed on its display (step 607).

21/5/4 (Item 1 from file: 350)
DIALOG(R)File 350:Derwent WPIX

016088164 **Image available**
WPI Acc No: 2004-246039/200423
Related WPI Acc No: 2004-614478
XRPX Acc No: N04-195059

Automatic cataloguing system for documents **located in multiple heterogeneous repositories, has portal server that retrieves user requests through a computer** network **and looks up information stored in metadata databases**

Patent Assignee: SCI APPL INT CORP (SCIT-N)
Inventor: ANTHONY D M; CONOVER J E

Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6701314	B1	20040302	US 2000489735	A	20000121	200423	B

Priority Applications (No Type Date): US 2000489735 A 20000121
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6701314	B1		12	G06F-007/00	

Abstract (Basic): US 6701314 B1

NOVELTY - A portal server (103) retrieves user requests through a computer **network** and looks up information stored in metadata databases. Metadata is encoded in an extensible mark-up language (XML)/resource description framework (RDF) format and stored in a delivery server to facilitate effective searching and retrieval of information from an information repository.

DETAILED DESCRIPTION - Metadata includes information including a **class** mark definition for each **document** .

An INDEPENDENT CLAIM is also included for a method in automatically cataloguing **documents** located in heterogeneous repositories.

USE - For **documents** located in multiple heterogeneous repositories.

ADVANTAGE - Provides a **search system** which can be used to perform search across many heterogeneous **information retrieval systems** , and can index and catalog information stored in many different formats on different websites, permitting users to perform a search through a single **web** portal. Uses a **thesaurus** and a classification system to determine both keywords for an indexed **document** but also a **class** for the **document** to permit more effective search and retrieval of information.

DESCRIPTION OF DRAWING(S) - The figure shows the automated cataloguing and index system.

pp; 12 DwgNo 2/5

Title Terms: AUTOMATIC; SYSTEM; **DOCUMENT** ; LOCATE; MULTIPLE; HETEROGENEOUS ; PORTAL; SERVE; RETRIEVAL; USER; REQUEST; THROUGH; COMPUTER; **NETWORK** ; UP; INFORMATION; STORAGE
Derwent Class: T01
International Patent Class (Main): **G06F-007/00**
File Segment: EPI

21/5/7 (Item 4 from file: 350)
DIALOG(R)File 350:Derwent WPIX

013576190 **Image available**
WPI Acc No: 2001-060397/200107
XRPX Acc No: N01-045195
Database document **search for building user defined technical** thesaurus **, involves displaying crafted query result obtained by modifying initial query via user interaction based on which** remote **database is searched**
Patent Assignee: PROCTER & GAMBLE CO (PROC)
Inventor: KIRKPATRICK J F; TOOGOOD K C
Number of Countries: 090 Number of Patents: 003
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
WO 200054185	A1	20000914	WO 2000US6072	A	20000308	200107	B
AU 200040070	A	20000928	AU 200040070	A	20000308	200107	
EP 1212697	A1	20020612	EP 2000919374	A	20000308	200239	
			WO 2000US6072	A	20000308		

Priority Applications (No Type Date): US 99264299 A 19990308
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
WO 200054185	A1	E	34	G06F-017/30	
AU 200040070	A			G06F-017/30	Based on patent WO 200054185
EP 1212697	A1	E		G06F-017/30	Based on patent WO 200054185

Designated States (National): AE AL AM AT AU AZ BA BB BG BR BY CA CH CN CR CU CZ DE DK DM EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX NO NZ PL PT RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG UZ VN YU ZA ZW
Designated States (Regional): AT BE CH CY DE DK EA ES FI FR GB GH GM GR IE IT KE LS LU MC MW NL OA PT SD SE SL SZ TZ UG ZW
Designated States (Regional): AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC

Abstract (Basic): WO 200054185 A1

NOVELTY - A user is iteratively prompted to interactively modify a chosen initial query to a crafted query, based on which at least one **remote** database is searched. The crafted query result is displayed on video monitor, depending on which the crafted query is modified again for search of required database.

DETAILED DESCRIPTION - The **remote** database comprises an electronically accessible bulk memory storage device that contains many technical reference **documents** . The initial query comprises at least one word or phrase and the crafted query comprises at least one word of phrase in addition to at least one equivalent word or phrase. The final version of crafted query comprises two terms that are formed into an expression using at least one **Boolean** operator. The initial and crafted query is associated with at least one image that is displayed on video monitor in a location proximal to **textual** information of initial and crafted query respectively. Multiple fields of information from at least one **online document** found in at least one **remote** database is linked together on sliding scale. At least one technical **thesaurus** using crafted query result is created. An INDEPENDENT CLAIM is also included for the **networked** computer **system** used for **searching documents** in database.

USE - For building **thesaurus** that are user defined in interactive data gathering system which access databases over **internet** .

ADVANTAGE - Searching of **on - line** technical **documents** is made easier by using interactively created query that uses both exact and equivalent search terms.

DESCRIPTION OF DRAWING(S) - The figure shows the flowchart of database **document** searching method involving **steps** of displaying and integrating resulting of search.

pp; 34 DwgNo 5/7

Title Terms: DATABASE; **DOCUMENT** ; SEARCH; BUILD; USER; DEFINE; TECHNICAL; DISPLAY; QUERY; RESULT; OBTAIN; MODIFIED; INITIAL; QUERY; USER; INTERACT; BASED; **REMOTE** ; DATABASE; SEARCH
Derwent Class: T01
International Patent Class (Main): **G06F-017/30**
File Segment: EPI

21/5/8 (Item 5 from file: 350)
DIALOG(R)File 350:Derwent WPIX

012899207 **Image available**
WPI Acc No: 2000-071042/200006
Related WPI Acc No: 1998-159050
XRPX Acc No: N00-055461
Program for intelligent search engine **in** fuzzy **logic system**
Patent Assignee: COMPAQ COMPUTER CORP (COPQ)
Inventor: NGUYEN T D
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week
US 5995956	A	19991130	US 9348880	A	19930416	200006 B
			US 95436502	A	19950508	
			US 97921218	A	19970827	

Priority Applications (No Type Date): US 9348880 A 19930416; US 95436502 A 19950508; US 97921218 A 19970827
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 5995956	A		10	G06F-015/18	Cont of application US 9348880 Cont of application US 95436502 Cont of patent US 5720001

Abstract (Basic): US 5995956 A
NOVELTY - A portion of computer program code written, receives natural language description comprising of one or more strings of alphanumeric characters of problem. Another portion of computer program code is used for searching questionless case-based knowledge base to identity questionless case structures that matches with the natural language description of problem.
DETAILED DESCRIPTION - A questionless case-based knowledge base comprising series of questionless case structures is formed from **on - line documentation** line. The **on - line documentation** comprises title, description of particular problem, solution or path to solution of particular problem. A computer program also ranks identified questionless case structures and displays titles of ranked identified questionless case structures on a display.
USE - For constructing knowledge base for intelligent **search engine** in **fuzzy** logic system.
ADVANTAGE - By constructing knowledge base from preexisting **on - line documentation** , reduction in manpower is achieved.
DESCRIPTION OF DRAWING(S) - The figure shows the flow chart of the **steps** involved in constructing questionless case based knowledge base.
pp; 10 DwgNo 3/4
Title Terms: PROGRAM; INTELLIGENCE; SEARCH; ENGINE; FUZZ; LOGIC; SYSTEM
Derwent Class: T01
International Patent Class (Main): **G06F-015/18**
File Segment: EPI

21/5/9 (Item 6 from file: 350)
DIALOG(R)File 350:Derwent WPIX

012773777 **Image available**
WPI Acc No: 1999-580004/199949
Related WPI Acc No: 2002-082048
XRPX Acc No: N99-428185

Database search representing method for graphical user interface for use in database management system

Patent Assignee: SZABO A J (SZAB-I)
Inventor: SZABO A J
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 5966126	A	19991012	US 96772650	A	19961223	199949	B

Priority Applications (No Type Date): US 96772650 A 19961223
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 5966126	A		34	G06F-003/00	

Abstract (Basic): US 5966126 A

NOVELTY - Minimum of three search selections, each for selecting records from a database, are received. Graphic icons are provided, for representing a composite set inclusion property of two selections. User selection of graphic icons, and the linkages between the icons, are received. The selected graphic icons and the linkages are displayed as a graphic image.

DETAILED DESCRIPTION - An INDEPENDENT CLAIM is also included for data set control system in GUI.

USE - For graphical user interface for use in database management system, to formulate and refine search.

ADVANTAGE - Provides a more intuitive language for the presentation and use of logical relationships between elements. Desired information can be analyzed and extracted, by defining a group of less complex rule sets and then consolidating the rule sets. Presents a basic or generic icon, defining several logical data regions representing a **Boolean** relationship between two or more data sets. Extends the standard **Boolean** logic with the known search operators to provide potentially full functionality within the logic representation. Efficiency may be gained in certain instance by providing functions with greater number of inputs and outputs, which are used as accessible icons. The use of **multidimensional** icons is avoided, so as to simplify the interface. Allows chaining of the binary representations in tree format, to achieve the complex results or transfer functions, thereby allowing the user to view the formulation of the search and to modify any element within the formulation, which can immediately update the entire search structure. Allows use of various optimized graphic representations of the underlying data sets and set operations, based on the preferences of the user and the context. Allows user to interact with multiple databases and **search engines** simultaneously, where each separate search is capable of being displayed by a separate Boole-graph. Offers the ability to search databases which include not only **textual** information, but also other information forms, such as auditory, visual, olfactory or touch inputs, etc.,

DESCRIPTION OF DRAWING(S) - The figure shows graphic search representations.

pp; 34 DwgNo 3/10

Title Terms: DATABASE; SEARCH; REPRESENT; METHOD; GRAPHICAL; USER; INTERFACE; DATABASE; MANAGEMENT; SYSTEM
Derwent Class: T01
International Patent Class (Main): **G06F-003/00**

File Segment: EPI

25/5/12 (Item 11 from file: 350)
DIALOG(R)File 350:Derwent WPIX

013455712 **Image available**
WPI Acc No: 2000-627655/200060
XRPX Acc No: N00-465000

Information retrieval system **using natural language queries in Internet, analyzes language based database and natural language query to generate database keywords and query keywords, respectively**

Patent Assignee: NOVELL INC (NOVE-N)
Inventor: AKKER D V D; DE BIE P; DE HITA C R; DEUN K V; GOVAERS E C E; LAVIOLETTE S; MACPHERSON M; PLATTEAU F M J
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 6081774	A	20000627	US 97916628	A	19970822	200060	B

Priority Applications (No Type Date): US 97916628 A 19970822
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 6081774	A		41	G06F-017/27	

Abstract (Basic): US 6081774 A

NOVELTY - A non-real time development system (102) and a real time retrieval system (104) **morphologically** , syntactically and **linguistically** analyze a language based database and natural language query, respectively to generate one or more database keywords and query keywords, respectively. The database and query keywords represent content of language based database and natural language query (160), respectively.

DETAILED DESCRIPTION - The non-real time development system creates a database index (130) having one or more content based keywords of the database, automatically. The real time retrieval **system searches** the index for query keywords derived from natural language query based on user's queries. The non-real time development system comprises a software developer's kit for creating database index, utilizing a pattern dictionary that includes **synonyms** and skipwords. A morphous syntactic dictionary in the system includes **morphological** and syntactic information for words in the natural language of language based database and natural language query. The real time retrieval system has a natural language interface (170) that creates one or more query keywords utilizing pattern and morphosyntactic dictionaries. A query index matcher matches one or more query keywords with one or more database keywords.

USE - For retrieving information from language based database using natural language queries in Internet and intranet.

ADVANTAGE - Enables any software developer to add **information retrieval system** to pre-existing software application to provide a user interface that enables user to develop a query in natural language. The software developer's kit enables software developers to add natural language interface and associated information retrieval capability to existing software application without any development work.

DESCRIPTION OF DRAWING(S) - The figure shows functional block diagram of **information retrieval system** .

Non-real time development system (102)
Real time retrieval system (104)
Database index (130)
Natural language query (160)
Natural language interface (170)
pp; 41 DwgNo 1/19

Title Terms: INFORMATION; RETRIEVAL; SYSTEM; NATURAL; LANGUAGE; QUERY; LANGUAGE; BASED; DATABASE; NATURAL; LANGUAGE; QUERY; GENERATE; DATABASE; KEYWORD; QUERY; KEYWORD; RESPECTIVE

25/5/18 (Item 17 from file: 350)
DIALOG(R)File 350:Derwent WPIX

003626441
WPI Acc No: 1983-H4643K/198322
XRPX Acc No: N83-097139
Data base searching system using variable search criteria - has field format register which specifies width and location of each field in record to define information format
Patent Assignee: SPERRY CORP (SPER)
Inventor: MANNING B W; PRECKSHOT N E; SLECHTA L J; WAGNER H M
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
US 4384325	A	19830517				198322	B

Priority Applications (No Type Date): US 80161983 A 19800623
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
US 4384325	A		20		

Abstract (Basic): US 4384325 A

The data base consists of a set of files or portions of them. Each file is divided into a number of records whereby all records of a given file have the same format but the records of different files may have different formats. A field format register is used to define the format of the records within a given file. The field format register specifies the location and width of each field within a record. To perform a search, a field-by-field comparison of each record is made to a reference word. The comparison yields a less than, equal to or greater than result for each field of each record.

A field comparison register describes the expected result of the field-by-field comparison. A given field is designated true if the comparison yields the expected result specified in the field comparison register. A hit on a given record is defined as satisfying a **Boolean** expression using the field-by-field true-false definitions as input variables. A given record is a miss if the **Boolean** expression is not satisfied.

3/15

Title Terms: DATA; BASE; SEARCH; SYSTEM; VARIABLE; SEARCH; CRITERIA; FIELD; FORMAT; REGISTER; SPECIFIED; WIDTH; LOCATE; FIELD; RECORD; DEFINE; INFORMATION; FORMAT
Derwent Class: T01
International Patent Class (Additional): **G06F-007/34**
File Segment: EPI

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page

Computer Science Department,
Stanford University, Stanford, CA 94305, USA
sergey@cs.stanford.edu and page@cs.stanford.edu

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/
To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine -- the first such detailed public description we know of to date. Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

Keywords

World Wide Web, Search Engines, Information Retrieval, PageRank, Google

1. Introduction

(Note: There are two versions of this paper -- a longer full version and a shorter printed version. The full version is available on the web and the conference CD-ROM.)
The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as Yahoo! or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or 10^{100} and fits well with our goal of building very large-scale search

engines.

1.1 Web Search Engines -- Scaling Up: 1994 - 2000

Search engine technology has had to scale dramatically to keep up with the growth of the web. In 1994, one of the first web search engines, the World Wide Web Worm (WWWW) [McBryan 94] had an index of 110,000 web pages and web accessible documents. As of November, 1997, the top search engines claim to index from 2 million (WebCrawler) to 100 million web documents (from Search Engine Watch). It is foreseeable that by the year 2000, a comprehensive index of the Web will contain over a billion documents. At the same time, the number of queries search engines handle has grown incredibly too. In March and April 1994, the World Wide Web Worm received an average of about 1500 queries per day. In November 1997, Altavista claimed it handled roughly 20 million queries per day. With the increasing number of users on the web, and automated systems which query search engines, it is likely that top search engines will handle hundreds of millions of queries per day by the year 2000. The goal of our system is to address many of the problems, both in quality and scalability, introduced by scaling search engine technology to such extraordinary numbers.

1.2. Google: Scaling with the Web

Creating a search engine which scales even to today's web presents many challenges. Fast crawling technology is needed to gather the web documents and keep them up to date. Storage space must be used efficiently to store indices and, optionally, the documents themselves. The indexing system must process hundreds of gigabytes of data efficiently. Queries must be handled quickly, at a rate of hundreds to thousands per second.

These tasks are becoming increasingly difficult as the Web grows. However, hardware performance and cost have improved dramatically to partially offset the difficulty. There are, however, several notable exceptions to this progress such as disk seek time and operating system robustness. In designing Google, we have considered both the rate of growth of the Web and technological changes. Google is designed to scale well to extremely large data sets. It makes efficient use of storage space to store the index. Its data structures are optimized for fast and efficient access (see section 4.2). Further, we expect that the cost to index and store text or HTML will eventually decline relative to the amount that will be available (see Appendix B). This will result in favorable scaling properties for centralized systems like Google.

1.3 Design Goals

1.3.1 Improved Search Quality

Our main goal is to improve the quality of web search engines. In 1994, some people believed that a complete search index would make it possible to find anything easily. According to Best of the Web 1994 -- Navigators, "The best navigation service should make it easy to find almost anything on the Web (once all the data is entered)." However, the Web of 1997 is quite different. Anyone who has used a search engine recently, can readily testify that the completeness of the index is not the only factor in the quality of search results. "Junk results" often wash out any results that a user is interested in. In fact, as of November 1997, only one of the top four commercial search engines finds itself (returns its own search page in response to its name in the top ten results). One of the main causes of this problem is that the number of documents in the indices has been increasing by many orders of magnitude, but the user's ability to look at documents has not. People are still only willing to look at the first few tens of results.

Because of this, as the collection size grows, we need tools that have very high precision (number of relevant documents returned, say in the top tens of results). Indeed, we want our notion of "relevant" to only include the very best documents since there may be tens of thousands of slightly relevant documents. This very high precision is important even at the expense of recall (the total number of relevant documents the system is able to return). There is quite a bit of recent optimism that the use of more hypertextual information can help improve search and other applications [Marchiori 97] [Spertus 97] [Weiss 96] [Kleinberg 98]. In particular, link structure [Page 98] and link text provide a lot of information for making relevance judgments and quality filtering. Google makes use of both link structure and anchor text (see Sections 2.1 and 2.2).

1.3.2 Academic Search Engine Research

Aside from tremendous growth, the Web has also become increasingly commercial over time. In 1993, 1.5% of web servers were on .com domains. This number grew to over 60% in 1997. At the same time, search engines have migrated from the academic domain to the commercial. Up until now most search engine development has gone on at companies with little publication of technical details. This causes search engine technology to remain largely a black art and to be advertising oriented (see Appendix A). With Google, we have a strong goal to push more development and understanding into the academic realm.

Another important design goal was to build systems that reasonable numbers of people can actually use. Usage was important to us because we think some of the most interesting research will involve leveraging the vast amount of usage data that is available from modern web systems. For example, there are many tens of millions of searches performed every day. However, it is very difficult to get this data, mainly because it is considered commercially valuable.

Our final design goal was to build an architecture that can support novel research activities on large-scale web data. To support novel research uses, Google stores all of the actual documents it crawls in compressed form. One of our main goals in designing Google was to set up an environment where other researchers can come in quickly, process large chunks of the web, and produce interesting results that would have been very difficult to produce otherwise. In the short time the system has been up, there have already been several papers using databases generated by Google, and many others are underway. Another goal we have is to set up a Spacelab-like environment where researchers or even students can propose and do interesting experiments on our large-scale web data.

2. System Features

The Google search engine has two important features that help it produce high precision results. First, it makes use of the link structure of the Web to calculate a quality ranking for each web page. This ranking is called PageRank and is described in detail in [Page 98]. Second, Google utilizes link to improve search results.

2.1 PageRank: Bringing Order to the Web

The citation (link) graph of the web is an important resource that has largely gone unused in existing web search engines. We have created maps containing as many as 518 million of these hyperlinks, a significant sample of the total. These maps allow rapid calculation of a web page's "PageRank", an

objective measure of its citation importance that corresponds well with people's subjective idea of importance. Because of this correspondence, PageRank is an excellent way to prioritize the results of web keyword searches. For most popular subjects, a simple text matching search that is restricted to web page titles performs admirably when PageRank prioritizes the results (demo available at google.stanford.edu). For the type of full text searches in the main Google system, PageRank also helps a great deal.

2.1.1 Description of PageRank Calculation

Academic citation literature has been applied to the web, largely by counting citations or backlinks to a given page. This gives some approximation of a page's importance or quality. PageRank extends this idea by not counting links from all pages equally, and by normalizing by the number of links on a page. PageRank is defined as follows:

> *We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:*
>
> $PR(A) = (1-d) + d\,(PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))$
>
> *Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one.*

PageRank or *PR(A)* can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web. Also, a PageRank for 26 million web pages can be computed in a few hours on a medium size workstation. There are many other details which are beyond the scope of this paper.

2.1.2 Intuitive Justification

PageRank can be thought of as a model of user behavior. We assume there is a "random surfer" who is given a web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its PageRank. And, the *d* damping factor is the probability at each page the "random surfer" will get bored and request another random page. One important variation is to only add the damping factor *d* to a single page, or a group of pages. This allows for personalization and can make it nearly impossible to deliberately mislead the system in order to get a higher ranking. We have several other extensions to PageRank, again see [Page 98].

Another intuitive justification is that a page can have a high PageRank if there are many pages that point to it, or if there are some pages that point to it and have a high PageRank. Intuitively, pages that are well cited from many places around the web are worth looking at. Also, pages that have perhaps only one citation from something like the Yahoo! homepage are also generally worth looking at. If a page was not high quality, or was a broken link, it is quite likely that Yahoo's homepage would not link to it. PageRank handles both these cases and everything in between by recursively propagating weights through the link structure of the web.

2.2 Anchor Text

The text of links is treated in a special way in our search engine. Most search engines associate the text of a link with the page that the link is on. In addition, we associate it with the page the link points to. This has several advantages. First, anchors often provide more accurate descriptions of web pages than the pages themselves. Second, anchors may exist for documents which cannot be indexed by a text-based search engine, such as images, programs, and databases. This makes it possible to return web pages which have not actually been crawled. Note that pages that have not been crawled can cause problems, since they are never checked for validity before being returned to the user. In this case, the search engine can even return a page that never actually existed, but had hyperlinks pointing to it. However, it is possible to sort the results, so that this particular problem rarely happens.

This idea of propagating anchor text to the page it refers to was implemented in the World Wide Web Worm [McBryan 94] especially because it helps search non-text information, and expands the search coverage with fewer downloaded documents. We use anchor propagation mostly because anchor text can help provide better quality results. Using anchor text efficiently is technically difficult because of the large amounts of data which must be processed. In our current crawl of 24 million pages, we had over 259 million anchors which we indexed.

2.3 Other Features

Aside from PageRank and the use of anchor text, Google has several other features. First, it has location information for all hits and so it makes extensive use of proximity in search. Second, Google keeps track of some visual presentation details such as font size of words. Words in a larger or bolder font are weighted higher than other words. Third, full raw HTML of pages is available in a repository.

3 Related Work

Search research on the web has a short and concise history. The World Wide Web Worm (WWWW) [McBryan 94] was one of the first web search engines. It was subsequently followed by several other academic search engines, many of which are now public companies. Compared to the growth of the Web and the importance of search engines there are precious few documents about recent search engines [Pinkerton 94]. According to Michael Mauldin (chief scientist, Lycos Inc) [Mauldin], "the various services (including Lycos) closely guard the details of these databases". However, there has been a fair amount of work on specific features of search engines. Especially well represented is work which can get results by post-processing the results of existing commercial search engines, or produce small scale "individualized" search engines. Finally, there has been a lot of research on information retrieval systems, especially on well controlled collections. In the next two sections, we discuss some areas where this research needs to be extended to work better on the web.

3.1 Information Retrieval

Work in information retrieval systems goes back many years and is well developed [Witten 94]. However, most of the research on information retrieval systems is on small well controlled homogeneous collections such as collections of scientific papers or news stories on a related topic. Indeed, the primary benchmark for information retrieval, the Text Retrieval Conference [TREC 96], uses a fairly small, well controlled collection for their benchmarks. The "Very Large Corpus"

benchmark is only 20GB compared to the 147GB from our crawl of 24 million web pages. Things that work well on TREC often do not produce good results on the web. For example, the standard vector space model tries to return the document that most closely approximates the query, given that both query and document are vectors defined by their word occurrence. On the web, this strategy often returns very short documents that are the query plus a few words. For example, we have seen a major search engine return a page containing only "Bill Clinton Sucks" and picture from a "Bill Clinton" query. Some argue that on the web, users should specify more accurately what they want and add more words to their query. We disagree vehemently with this position. If a user issues a query like "Bill Clinton" they should get reasonable results since there is a enormous amount of high quality information available on this topic. Given examples like these, we believe that the standard information retrieval work needs to be extended to deal effectively with the web.

3.2 Differences Between the Web and Well Controlled Collections

The web is a vast collection of completely uncontrolled heterogeneous documents. Documents on the web have extreme variation internal to the documents, and also in the external meta information that might be available. For example, documents differ internally in their language (both human and programming), vocabulary (email addresses, links, zip codes, phone numbers, product numbers), type or format (text, HTML, PDF, images, sounds), and may even be machine generated (log files or output from a database). On the other hand, we define external meta information as information that can be inferred about a document, but is not contained within it. Examples of external meta information include things like reputation of the source, update frequency, quality, popularity or usage, and citations. Not only are the possible sources of external meta information varied, but the things that are being measured vary many orders of magnitude as well. For example, compare the usage information from a major homepage, like Yahoo's which currently receives millions of page views every day with an obscure historical article which might receive one view every ten years. Clearly, these two items must be treated very differently by a search engine.

Another big difference between the web and traditional well controlled collections is that there is virtually no control over what people can put on the web. Couple this flexibility to publish anything with the enormous influence of search engines to route traffic and companies which deliberately manipulating search engines for profit become a serious problem. This problem that has not been addressed in traditional closed information retrieval systems. Also, it is interesting to note that metadata efforts have largely failed with web search engines, because any text on the page which is not directly represented to the user is abused to manipulate search engines. There are even numerous companies which specialize in manipulating search engines for profit.

4 System Anatomy

First, we will provide a high level discussion of the architecture. Then, there is some in-depth descriptions of important data structures. Finally, the major applications: crawling, indexing, and searching will be examined in depth.

4.1 Google Architecture Overview

In this section, we will give a high level overview of how the whole system works as pictured in Figure 1. Further sections will discuss the applications and data structures not mentioned in this section. Most of Google is implemented in C or C++ for efficiency and can run in either Solaris or Linux.

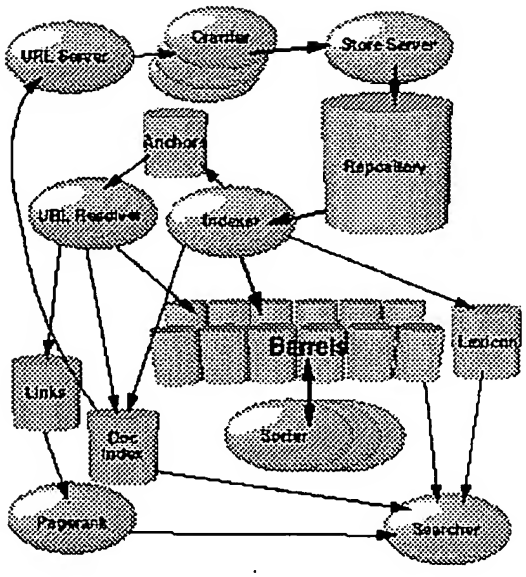


Figure 1. High Level Google Architecture

In Google, the web crawling (downloading of web pages) is done by several distributed crawlers. There is a URLserver that sends lists of URLs to be fetched to the crawlers. The web pages that are fetched are then sent to the storeserver. The storeserver then compresses and stores the web pages into a repository. Every web page has an associated ID number called a docID which is assigned whenever a new URL is parsed out of a web page. The indexing function is performed by the indexer and the sorter. The indexer performs a number of functions. It reads the repository, uncompresses the documents, and parses them. Each document is converted into a set of word occurrences called hits. The hits record the word, position in document, an approximation of font size, and capitalization. The indexer distributes these hits into a set of "barrels", creating a partially sorted forward index. The indexer performs another important function. It parses out all the links in every web page and stores important information about them in an anchors file. This file contains enough information to determine where each link points from and to, and the text of the link.

The URLresolver reads the anchors file and converts relative URLs into absolute URLs and in turn into docIDs. It puts the anchor text into the forward index, associated with the docID that the anchor points to. It also generates a database of links which are pairs of docIDs. The links database is used to compute PageRanks for all the documents.

The sorter takes the barrels, which are sorted by docID (this is a simplification, see Section 4.2.5), and resorts them by wordID to generate the inverted index. This is done in place so that little temporary space is needed for this operation. The sorter also produces a list of wordIDs and offsets into the inverted index. A program called DumpLexicon takes this list together with the lexicon produced by the indexer and generates a new lexicon to be used by the searcher. The searcher is run by a web server and uses the lexicon built by DumpLexicon together with the inverted index and the PageRanks to answer queries.

4.2 Major Data Structures

Google's data structures are optimized so that a large document collection can be crawled, indexed, and searched with little cost. Although, CPUs and bulk input output rates have improved dramatically over the years, a disk seek still requires about 10 ms to complete. Google is designed to avoid disk seeks whenever possible, and this has had a considerable influence on the design of the data structures.

4.2.1 BigFiles

BigFiles are virtual files spanning multiple file systems and are addressable by 64 bit integers. The allocation among multiple file systems is handled automatically. The BigFiles package also handles allocation and deallocation of file descriptors, since the operating systems do not provide enough for our needs. BigFiles also support rudimentary compression options.

4.2.2 Repository

The repository contains the full HTML of every web page. Each page is compressed using zlib (see RFC1950). The choice of compression technique is a tradeoff between speed and compression ratio. We chose zlib's speed over a significant improvement in compression offered by bzip. The compression rate of bzip was approximately 4 to 1 on the repository as compared to zlib's 3 to 1 compression. In the repository, the documents are stored one after the other and are prefixed by docID, length, and URL as can be seen in Figure 2. The repository requires no other data structures to be used in order to access it. This helps with data consistency and makes development much easier; we can rebuild all the other data structures from only the repository and a file which lists crawler errors.

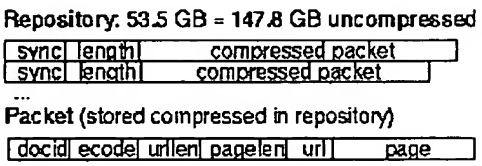


Figure 2. Repository Data Structure

4.2.3 Document Index

The document index keeps information about each document. It is a fixed width ISAM (Index sequential access mode) index, ordered by docID. The information stored in each entry includes the current document status, a pointer into the repository, a document checksum, and various statistics. If the document has been crawled, it also contains a pointer into a variable width file called docinfo which contains its URL and title. Otherwise the pointer points into the URLlist which contains just the URL. This design decision was driven by the desire to have a reasonably compact data structure, and the ability to fetch a record in one disk seek during a search

Additionally, there is a file which is used to convert URLs into docIDs. It is a list of URL checksums with their corresponding docIDs and is sorted by checksum. In order to find the docID of a particular URL, the URL's checksum is computed and a binary search is performed on the checksums file to find its docID. URLs may be converted into docIDs in batch by doing a merge with this file. This is the technique the URLresolver uses to turn URLs into docIDs. This batch mode of update is crucial because otherwise we must perform one seek for every link which assuming one disk would take more than a month for our 322 million link dataset.

4.2.4 Lexicon

The lexicon has several different forms. One important change from earlier systems is that the lexicon can fit in memory for a reasonable price. In the current implementation we can keep the lexicon in memory on a machine with 256 MB of main memory. The current lexicon contains 14 million words (though some rare words were not added to the lexicon). It is implemented in two parts -- a list of the words (concatenated together but separated by nulls) and a hash table of pointers. For various functions,

the list of words has some auxiliary information which is beyond the scope of this paper to explain fully.

4.2.5 Hit Lists

A hit list corresponds to a list of occurrences of a particular word in a particular document including position, font, and capitalization information. Hit lists account for most of the space used in both the forward and the inverted indices. Because of this, it is important to represent them as efficiently as possible. We considered several alternatives for encoding position, font, and capitalization -- simple encoding (a triple of integers), a compact encoding (a hand optimized allocation of bits), and Huffman coding. In the end we chose a hand optimized compact encoding since it required far less space than the simple encoding and far less bit manipulation than Huffman coding. The details of the hits are shown in Figure 3.

Our compact encoding uses two bytes for every hit. There are two types of hits: fancy hits and plain hits. Fancy hits include hits occurring in a URL, title, anchor text, or meta tag. Plain hits include everything else. A plain hit consists of a capitalization bit, font size, and 12 bits of word position in a document (all positions higher than 4095 are labeled 4096). Font size is represented relative to the rest of the document using three bits (only 7 values are actually used because 111 is the flag that signals a fancy hit). A fancy hit consists of a capitalization bit, the font size set to 7 to indicate it is a fancy hit, 4 bits to encode the type of fancy hit, and 8 bits of position. For anchor hits, the 8 bits of position are split into 4 bits for position in anchor and 4 bits for a hash of the docID the anchor occurs in. This gives us some limited phrase searching as long as there are not that many anchors for a particular word. We expect to update the way that anchor hits are stored to allow for greater resolution in the position and docIDhash fields. We use font size relative to the rest of the document because when searching, you do not want to rank otherwise identical documents differently just because one of the documents is in a larger font.

The length of a hit list is stored before the hits themselves. To save space, the length of the hit list is combined with the wordID in the forward index and the docID in the inverted index. This limits it to 8 and 5 bits respectively (there are some tricks which allow 8 bits to be borrowed from the wordID). If the length is longer than would fit in that many bits, an escape code is used in those bits, and the next two bytes contain the actual length.

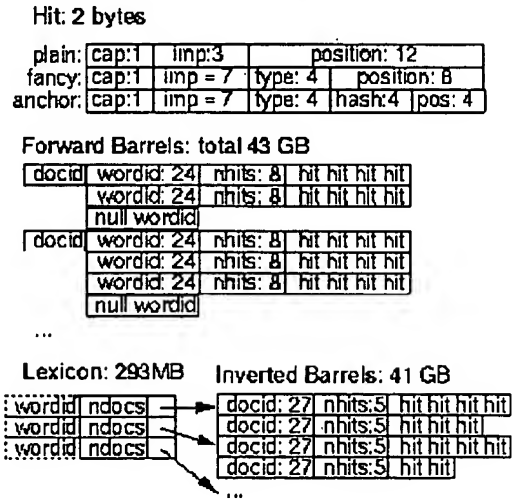


Figure 3. Forward and Reverse Indexes and the Lexicon

4.2.6 Forward Index

The forward index is actually already partially sorted. It is stored in a number of barrels (we used 64). Each barrel holds a range of wordID's. If a document contains words that fall into a particular barrel, the docID is recorded into the barrel, followed by a list of wordID's with hitlists which correspond to those words. This scheme requires slightly more storage because of duplicated docIDs but the difference is very small for a reasonable number of buckets and saves considerable time and coding complexity in the final indexing phase done by the sorter. Furthermore, instead of storing actual wordID's, we store each wordID as a relative difference from the minimum wordID that falls into the

barrel the wordID is in. This way, we can use just 24 bits for the wordID's in the unsorted barrels, leaving 8 bits for the hit list length.

4.2.7 Inverted Index

The inverted index consists of the same barrels as the forward index, except that they have been processed by the sorter. For every valid wordID, the lexicon contains a pointer into the barrel that wordID falls into. It points to a doclist of docID's together with their corresponding hit lists. This doclist represents all the occurrences of that word in all documents.

An important issue is in what order the docID's should appear in the doclist. One simple solution is to store them sorted by docID. This allows for quick merging of different doclists for multiple word queries. Another option is to store them sorted by a ranking of the occurrence of the word in each document. This makes answering one word queries trivial and makes it likely that the answers to multiple word queries are near the start. However, merging is much more difficult. Also, this makes development much more difficult in that a change to the ranking function requires a rebuild of the index. We chose a compromise between these options, keeping two sets of inverted barrels -- one set for hit lists which include title or anchor hits and another set for all hit lists. This way, we check the first set of barrels first and if there are not enough matches within those barrels we check the larger ones.

4.3 Crawling the Web

Running a web crawler is a challenging task. There are tricky performance and reliability issues and even more importantly, there are social issues. Crawling is the most fragile application since it involves interacting with hundreds of thousands of web servers and various name servers which are all beyond the control of the system.

In order to scale to hundreds of millions of web pages, Google has a fast distributed crawling system. A single URLserver serves lists of URLs to a number of crawlers (we typically ran about 3). Both the URLserver and the crawlers are implemented in Python. Each crawler keeps roughly 300 connections open at once. This is necessary to retrieve web pages at a fast enough pace. At peak speeds, the system can crawl over 100 web pages per second using four crawlers. This amounts to roughly 600K per second of data. A major performance stress is DNS lookup. Each crawler maintains a its own DNS cache so it does not need to do a DNS lookup before crawling each document. Each of the hundreds of connections can be in a number of different states: looking up DNS, connecting to host, sending request, and receiving response. These factors make the crawler a complex component of the system. It uses asynchronous IO to manage events, and a number of queues to move page fetches from state to state.

It turns out that running a crawler which connects to more than half a million servers, and generates tens of millions of log entries generates a fair amount of email and phone calls. Because of the vast number of people coming on line, there are always those who do not know what a crawler is, because this is the first one they have seen. Almost daily, we receive an email something like, "Wow, you looked at a lot of pages from my web site. How did you like it?" There are also some people who do not know about the robots exclusion protocol, and think their page should be protected from indexing by a statement like, "This page is copyrighted and should not be indexed", which needless to say is difficult for web crawlers to understand. Also, because of the huge amount of data involved, unexpected things will happen. For example, our system tried to crawl an online game. This resulted in lots of garbage messages in the middle of their game! It turns out this was an easy problem to fix. But this problem had not come up

until we had downloaded tens of millions of pages. Because of the immense variation in web pages and servers, it is virtually impossible to test a crawler without running it on large part of the Internet. Invariably, there are hundreds of obscure problems which may only occur on one page out of the whole web and cause the crawler to crash, or worse, cause unpredictable or incorrect behavior. Systems which access large parts of the Internet need to be designed to be very robust and carefully tested. Since large complex systems such as crawlers will invariably cause problems, there needs to be significant resources devoted to reading the email and solving these problems as they come up.

4.4 Indexing the Web

- **Parsing** – Any parser which is designed to run on the entire Web must handle a huge array of possible errors. These range from typos in HTML tags to kilobytes of zeros in the middle of a tag, non-ASCII characters, HTML tags nested hundreds deep, and a great variety of other errors that challenge anyone's imagination to come up with equally creative ones. For maximum speed, instead of using YACC to generate a CFG parser, we use flex to generate a lexical analyzer which we outfit with its own stack. Developing this parser which runs at a reasonable speed and is very robust involved a fair amount of work.
- **Indexing Documents into Barrels** – After each document is parsed, it is encoded into a number of barrels. Every word is converted into a wordID by using an in-memory hash table -- the lexicon. New additions to the lexicon hash table are logged to a file. Once the words are converted into wordID's, their occurrences in the current document are translated into hit lists and are written into the forward barrels. The main difficulty with parallelization of the indexing phase is that the lexicon needs to be shared. Instead of sharing the lexicon, we took the approach of writing a log of all the extra words that were not in a base lexicon, which we fixed at 14 million words. That way multiple indexers can run in parallel and then the small log file of extra words can be processed by one final indexer.
- **Sorting** -- In order to generate the inverted index, the sorter takes each of the forward barrels and sorts it by wordID to produce an inverted barrel for title and anchor hits and a full text inverted barrel. This process happens one barrel at a time, thus requiring little temporary storage. Also, we parallelize the sorting phase to use as many machines as we have simply by running multiple sorters, which can process different buckets at the same time. Since the barrels don't fit into main memory, the sorter further subdivides them into baskets which do fit into memory based on wordID and docID. Then the sorter, loads each basket into memory, sorts it and writes its contents into the short inverted barrel and the full inverted barrel.

4.5 Searching

The goal of searching is to provide quality search results efficiently. Many of the large commercial search engines seemed to have made great progress in terms of efficiency. Therefore, we have focused more on quality of search in our research, although we believe our solutions are scalable to commercial volumes with a bit more effort. The google query evaluation process is show in Figure 4.

To put a limit on response time, once a certain number (currently 40,000) of matching documents are found, the searcher automatically goes to step 8 in Figure 4. This means that it is possible that sub-optimal results would be returned. We are currently investigating other ways to solve this problem. In the past, we sorted the hits according to PageRank, which seemed to improve the situation.

1. Parse the query.
2. Convert words into wordIDs.
3. Seek to the start of the doclist in the short barrel for every word.
4. Scan through the doclists until there is a document that matches all the search terms.
5. Compute the rank of that document for the query.
6. If we are in the short barrels and at the end of any doclist, seek to the start of the doclist in the full barrel for every word and go to step 4.
7. If we are not at the end of any doclist go to step 4.
   Sort the documents that have matched by rank and return the top k.

Figure 4. Google Query Evaluation

4.5.1 The Ranking System

Google maintains much more information about web documents than typical search engines. Every hitlist includes position, font, and capitalization information. Additionally, we factor in hits from anchor text and the PageRank of the document. Combining all of this information into a rank is difficult. We designed our ranking function so that no particular factor can have too much influence. First, consider the simplest case -- a single word query. In order to rank a document with a single word query, Google looks at that document's hit list for that word. Google considers each hit to be one of several different types (title, anchor, URL, plain text large font, plain text small font, ...), each of which has its own type-weight. The type-weights make up a vector indexed by type. Google counts the number of hits of each type in the hit list. Then every count is converted into a count-weight. Count-weights increase linearly with counts at first but quickly taper off so that more than a certain count will not help. We take the dot product of the vector of count-weights with the vector of type-weights to compute an IR score for the document. Finally, the IR score is combined with PageRank to give a final rank to the document.

For a multi-word search, the situation is more complicated. Now multiple hit lists must be scanned through at once so that hits occurring close together in a document are weighted higher than hits occurring far apart. The hits from the multiple hit lists are matched up so that nearby hits are matched together. For every matched set of hits, a proximity is computed. The proximity is based on how far apart the hits are in the document (or anchor) but is classified into 10 different value "bins" ranging from a phrase match to "not even close". Counts are computed not only for every type of hit but for every type and proximity. Every type and proximity pair has a type-prox-weight. The counts are converted into count-weights and we take the dot product of the count-weights and the type-prox-weights to compute an IR score. All of these numbers and matrices can all be displayed with the search results using a special debug mode. These displays have been very helpful in developing the ranking system.

4.5.2 Feedback

The ranking function has many parameters like the type-weights and the type-prox-weights. Figuring out the right values for these parameters is something of a black art. In order to do this, we have a user feedback mechanism in the search engine. A trusted user may optionally evaluate all of the results that are returned. This feedback is saved. Then when we modify the ranking function, we can see the impact of this change on all previous searches which were ranked. Although far from perfect, this gives us some

idea of how a change in the ranking function affects the search results.

5 Results and Performance

The most important measure of a search engine is the quality of its search results. While a complete user evaluation is beyond the scope of this paper, our own experience with Google has shown it to produce better results than the major commercial search engines for most searches. As an example which illustrates the use of PageRank, anchor text, and proximity, Figure 4 shows Google's results for a search on "bill clinton". These results demonstrates some of Google's features. The results are clustered by server. This helps considerably when sifting through result sets. A number of results are from the whitehouse.gov domain which is what one may reasonably expect from such a search. Currently, most major commercial search engines do not return any results from whitehouse.gov, much less the right ones. Notice that there is no title for the first result. This is because it was not crawled. Instead, Google relied on anchor text to determine this was a good answer to the query. Similarly, the fifth result is an email address which, of course, is not crawlable. It is also a result of anchor text.

Query: bill clinton
http://www.whitehouse.gov/
100.00% (no date) (0K)
http://www.whitehouse.gov/
Office of the President
99.67% (Dec 23 1996) (2K)
http://www.whitehouse.gov/WH/EOP/OP/html/OP_Home.html
Welcome To The White House
99.98% (Nov 09 1997) (5K)
http://www.whitehouse.gov/WH/Welcome.html
Send Electronic Mail to the President
99.86% (Jul 14 1997) (5K)
http://www.whitehouse.gov/WH/Mail/html/Mail_President.html

mailto:president@whitehouse.gov
99.98%
mailto:President@whitehouse.gov
99.27%
The "Unofficial" Bill Clinton
94.06% (Nov 11 1997) (14K)
http://zpub.com/un/un-bc.html
Bill Clinton Meets The Shrinks
86.27% (Jun 29 1997) (63K)
http://zpub.com/un/un-bc9.html
President Bill Clinton - The Dark Side
97.27% (Nov 10 1997) (15K)
http://www.realchange.org/clinton.htm
$3 Bill Clinton
94.73% (no date) (4K)
http://www.gatewy.net/~tjohnson/clinton1.html

Figure 4. Sample Results from Google

All of the results are reasonably high quality pages and, at last check, none were broken links. This is largely because they all have high PageRank. The PageRanks are the percentages in red along with bar graphs. Finally, there are no results about a Bill other than Clinton or about a Clinton other than Bill. This is because we place heavy importance on the proximity of word occurrences. Of course a true test of the quality of a search engine would involve an extensive user study or results analysis which we do not have room for here. Instead, we invite the reader to try Google for themselves at http://google.stanford.edu.

5.1 Storage Requirements

Aside from search quality, Google is designed to scale cost effectively to the size of the Web as it grows. One aspect of this is to use storage efficiently. Table 1 has a breakdown of some statistics and storage requirements of Google. Due to compression the total size of the repository is about 53 GB, just over one third of the total data it stores. At current disk prices this makes the repository a relatively cheap source of useful data. More importantly, the total of all the data used by the search engine requires a comparable amount of storage, about 55 GB. Furthermore, most queries can be answered using just the short inverted index. With better encoding and compression of the Document Index, a high quality web search engine may fit onto a 7GB drive of a new PC.

5.2 System Performance

It is important for a search engine to crawl and index efficiently. This way information can be kept up to date and major changes to the system can be tested relatively quickly. For Google, the major operations are Crawling, Indexing, and Sorting. It is difficult to measure how long crawling took overall because disks filled up, name servers crashed, or any number of other problems which stopped the system. In total it took roughly 9 days to download the 26 million pages (including errors). However, once the system was running smoothly, it ran much faster, downloading the last 11 million pages in just 63 hours, averaging just over 4 million pages per day or 48.5 pages per second. We ran the indexer and the crawler simultaneously. The indexer ran just faster than the crawlers. This is largely because we spent just enough time optimizing the indexer so that it would not be a bottleneck. These optimizations included bulk updates to the document index and placement of critical data structures on the local disk. The indexer runs at roughly 54 pages per second. The sorters can be run completely in parallel; using four machines, the whole process of sorting takes about 24 hours.

Storage Statistics	
Total Size of Fetched Pages	147.8 GB
Compressed Repository	53.5 GB
Short Inverted Index	4.1 GB
Full Inverted Index	37.2 GB
Lexicon	293 MB
Temporary Anchor Data (not in total)	6.6 GB
Document Index Incl. Variable Width Data	9.7 GB
Links Database	3.9 GB
Total Without Repository	**55.2 GB**
Total With Repository	**108.7 GB**

Web Page Statistics	
Number of Web Pages Fetched	24 million
Number of Urls Seen	76.5 million
Number of Email Addresses	1.7 million
Number of 404's	1.6 million

Table 1. Statistics

5.3 Search Performance

Improving the performance of search was not the major focus of our research up to this point. The current version of Google answers most queries in between 1 and 10 seconds. This time is mostly dominated by disk IO over NFS (since disks are spread over a number of machines). Furthermore, Google does not have any optimizations such as query caching, subindices on common terms, and other common optimizations. We intend to speed up Google considerably through distribution and hardware, software, and algorithmic improvements. Our target is to be able to handle several hundred queries per second. Table 2 has some sample query times from the current version of Google. They are repeated to show the speedups resulting from cached IO.

6 Conclusions

Google is designed to be a scalable search engine. The primary goal is to provide high quality search results over a rapidly growing World Wide Web. Google employs a number of techniques to improve search quality including page rank, anchor text, and proximity information. Furthermore, Google is a complete architecture for gathering web pages, indexing them, and performing search queries over them.

	Initial Query		Same Query Repeated (IO mostly cached)	
Query	CPU Time(s)	Total Time(s)	CPU Time(s)	Total Time(s)
al gore	0.09	2.13	0.06	0.06
vice president	1.77	3.84	1.66	1.80
hard disks	0.25	4.86	0.20	0.24
search engines	1.31	9.63	1.16	1.16

Table 2. Search Times

6.1 Future Work

A large-scale web search engine is a complex system and much remains to be done. Our immediate goals are to improve search efficiency and to scale to approximately 100 million web pages. Some simple improvements to efficiency include query caching, smart disk allocation, and subindices. Another area which requires much research is updates. We must have smart algorithms to decide what old web pages should be recrawled and what new ones should be crawled. Work toward this goal has been done in [Cho 98]. One promising area of research is using proxy caches to build search databases, since they are demand driven. We are planning to add simple features supported by commercial search engines like boolean operators, negation, and stemming. However, other features are just starting to be explored such as relevance feedback and clustering (Google currently supports a simple hostname based clustering). We also plan to support user context (like the user's location), and result summarization. We are also working to extend the use of link structure and link text. Simple experiments indicate PageRank can be personalized by increasing the weight of a user's home page or bookmarks. As for link text, we are experimenting with using text surrounding links in addition to the link text itself. A Web search engine is a very rich environment for research ideas. We have far too many to list here so we do not expect this Future Work section to become much shorter in the near future.

6.2 High Quality Search

The biggest problem facing users of web search engines today is the quality of the results they get back. While the results are often amusing and expand users' horizons, they are often frustrating and consume precious time. For example, the top result for a search for "Bill Clinton" on one of the most popular commercial search engines was the Bill Clinton Joke of the Day: April 14, 1997. Google is designed to provide higher quality search so as the Web continues to grow rapidly, information can be found easily. In order to accomplish this Google makes heavy use of hypertextual information consisting of link structure and link (anchor) text. Google also uses proximity and font information. While evaluation of a search engine is difficult, we have subjectively found that Google returns higher quality search results than current commercial search engines. The analysis of link structure via PageRank allows Google to evaluate the quality of web pages. The use of link text as a description of what the link points to helps the search engine return relevant (and to some degree high quality) results. Finally, the use of proximity information helps increase relevance a great deal for many queries.

6.3 Scalable Architecture

Aside from the quality of search, Google is designed to scale. It must be efficient in both space and time, and constant factors are very important when dealing with the entire Web. In implementing Google, we have seen bottlenecks in CPU, memory access, memory capacity, disk seeks, disk throughput, disk capacity, and network IO. Google has evolved to overcome a number of these bottlenecks during various operations. Google's major data structures make efficient use of available storage space. Furthermore, the crawling, indexing, and sorting operations are efficient enough to be able to build an index of a substantial portion of the web -- 24 million pages, in less than one week. We expect to be able to build an index of 100 million pages in less than a month.

6.4 A Research Tool

In addition to being a high quality search engine, Google is a research tool. The data Google has collected has already resulted in many other papers submitted to conferences and many more on the way. Recent research such as [Abiteboul 97] has shown a number of limitations to queries about the Web that may be answered without having the Web available locally. This means that Google (or a similar system) is not only a valuable research tool but a necessary one for a wide range of applications. We hope Google will be a resource for searchers and researchers all around the world and will spark the next generation of search engine technology.

7 Acknowledgments

Scott Hassan and Alan Steremberg have been critical to the development of Google. Their talented contributions are irreplaceable, and the authors owe them much gratitude. We would also like to thank Hector Garcia-Molina, Rajeev Motwani, Jeff Ullman, and Terry Winograd and the whole WebBase group for their support and insightful discussions. Finally we would like to recognize the generous support of our equipment donors IBM, Intel, and Sun and our funders. The research described here was conducted as part of the Stanford Integrated Digital Library Project, supported by the National Science Foundation under Cooperative Agreement IRI-9411306. Funding for this cooperative agreement is also provided by DARPA and NASA, and by Interval Research, and the industrial partners of the Stanford Digital Libraries Project.

References

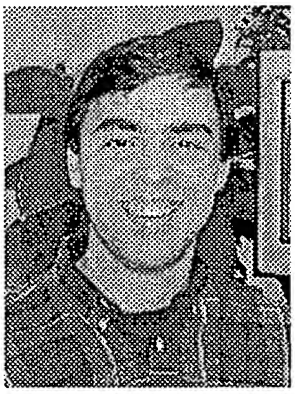
- Best of the Web 1994 -- Navigators http://botw.org/1994/awards/navigators.html
- Bill Clinton Joke of the Day: April 14, 1997. http://www.io.com/~cjburke/clinton/970414.html.
- Bzip2 Homepage http://www.muraroa.demon.co.uk/
- Google Search Engine http://google.stanford.edu/
- Harvest http://harvest.transarc.com/
- Mauldin, Michael L. Lycos Design Choices in an Internet Search Service, IEEE Expert Interview http://www.computer.org/pubs/expert/1997/trends/x1008/mauldin.htm
- The Effect of Cellular Phone Use Upon Driver Attention http://www.webfirst.com/aaa/text/cell/cell0toc.htm
- Search Engine Watch http://www.searchenginewatch.com/
- RFC 1950 (zlib) ftp://ftp.uu.net/graphics/png/documents/zlib/zdoc-index.html
- Robots Exclusion Protocol: http://info.webcrawler.com/mak/projects/robots/exclusion.htm

- Web Growth Summary: http://www.mit.edu/people/mkgray/net/web-growth-summary.html
- Yahoo! http://www.yahoo.com/

- [Abiteboul 97] Serge Abiteboul and Victor Vianu, *Queries and Computation on the Web.* Proceedings of the International Conference on Database Theory. Delphi, Greece 1997.
- [Bagdikian 97] Ben H. Bagdikian. *The Media Monopoly.* 5th Edition. Publisher: Beacon, ISBN: 0807061557
- [Cho 98] Junghoo Cho, Hector Garcia-Molina, Lawrence Page. *Efficient Crawling Through URL Ordering.* Seventh International Web Conference (WWW 98). Brisbane, Australia, April 14-18, 1998.
- [Gravano 94] Luis Gravano, Hector Garcia-Molina, and A. Tomasic. *The Effectiveness of GlOSS for the Text-Database Discovery Problem.* Proc. of the 1994 ACM SIGMOD International Conference On Management Of Data, 1994.
- [Kleinberg 98] Jon Kleinberg, *Authoritative Sources in a Hyperlinked Environment*, Proc. ACM-SIAM Symposium on Discrete Algorithms, 1998.
- [Marchiori 97] Massimo Marchiori. *The Quest for Correct Information on the Web: Hyper Search Engines.* The Sixth International WWW Conference (WWW 97). Santa Clara, USA, April 7-11, 1997.
- [McBryan 94] Oliver A. McBryan. GENVL and *WWWW: Tools for Taming the Web. First International Conference on the World Wide Web.* CERN, Geneva (Switzerland), May 25-26-27 1994. http://www.cs.colorado.edu/home/mcbryan/mypapers/www94.ps
- [Page 98] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.* Manuscript in progress. http://google.stanford.edu/~backrub/pageranksub.ps
- [Pinkerton 94] Brian Pinkerton, *Finding What People Want: Experiences with the WebCrawler.* The Second International WWW Conference Chicago, USA, October 17-20, 1994. http://info.webcrawler.com/bp/WWW94.html
- [Spertus 97] Ellen Spertus. *ParaSite: Mining Structural Information on the Web.* The Sixth International WWW Conference (WWW 97). Santa Clara, USA, April 7-11, 1997.
- [TREC 96] *Proceedings of the fifth Text REtrieval Conference (TREC-5).* Gaithersburg, Maryland, November 20-22, 1996. Publisher: Department of Commerce, National Institute of Standards and Technology. Editors: D. K. Harman and E. M. Voorhees. Full text at: http://trec.nist.gov/
- [Witten 94] Ian H Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* New York: Van Nostrand Reinhold, 1994.
- [Weiss 96] Ron Weiss, Bienvenido Velez, Mark A. Sheldon, Chanathip Manprempre, Peter Szilagyi, Andrzej Duda, and David K. Gifford. *HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering.* Proceedings of the 7th ACM Conference on Hypertext. New York, 1996.

Vitae

Sergey Brin received his B.S. degree in mathematics and computer science from the University of Maryland at College Park in 1993. Currently, he is a Ph.D. candidate in computer science at Stanford University where he received his M.S. in 1995. He is a recipient of a National Science Foundation Graduate Fellowship. His research interests include search engines, information extraction from unstructured sources, and data mining of large text collections and scientific data.



Lawrence Page was born in East Lansing, Michigan, and received a B.S.E. in Computer Engineering at the University of Michigan Ann Arbor in 1995. He is currently a Ph.D. candidate in Computer Science at Stanford University. Some of his research interests include the link structure of the web, human computer interaction, search engines, scalability of information access interfaces, and personal data mining.

8 Appendix A: Advertising and Mixed Motives

Currently, the predominant business model for commercial search engines is advertising. The goals of the advertising business model do not always correspond to providing quality search to users. For example, in our prototype search engine one of the top results for cellular phone is "The Effect of Cellular Phone Use Upon Driver Attention", a study which explains in great detail the distractions and risk associated with conversing on a cell phone while driving. This search result came up first because of its high importance as judged by the PageRank algorithm, an approximation of citation importance on the web [Page, 98]. It is clear that a search engine which was taking money for showing cellular phone ads would have difficulty justifying the page that our system returned to its paying advertisers. For this type of reason and historical experience with other media [Bagdikian 83], we expect that advertising funded search engines will be inherently biased towards the advertisers and away from the needs of the consumers.

Since it is very difficult even for experts to evaluate search engines, search engine bias is particularly insidious. A good example was OpenText, which was reported to be selling companies the right to be listed at the top of the search results for particular queries [Marchiori 97]. This type of bias is much more insidious than advertising, because it is not clear who "deserves" to be there, and who is willing to pay money to be listed. This business model resulted in an uproar, and OpenText has ceased to be a viable search engine. But less blatant bias are likely to be tolerated by the market. For example, a search engine could add a small factor to search results from "friendly" companies, and subtract a factor from results from competitors. This type of bias is very difficult to detect but could still have a significant effect on the market. Furthermore, advertising income often provides an incentive to provide poor

quality search results. For example, we noticed a major search engine would not return a large airline's homepage when the airline's name was given as a query. It so happened that the airline had placed an expensive ad, linked to the query that was its name. A better search engine would not have required this ad, and possibly resulted in the loss of the revenue from the airline to the search engine. In general, it could be argued from the consumer point of view that the better the search engine is, the fewer advertisements will be needed for the consumer to find what they want. This of course erodes the advertising supported business model of the existing search engines. However, there will always be money from advertisers who want a customer to switch products, or have something that is genuinely new. But we believe the issue of advertising causes enough mixed incentives that it is crucial to have a competitive search engine that is transparent and in the academic realm.

9 Appendix B: Scalability

9. 1 Scalability of Google

We have designed Google to be scalable in the near term to a goal of 100 million web pages. We have just received disk and machines to handle roughly that amount. All of the time consuming parts of the system are parallelize and roughly linear time. These include things like the crawlers, indexers, and sorters. We also think that most of the data structures will deal gracefully with the expansion. However, at 100 million web pages we will be very close up against all sorts of operating system limits in the common operating systems (currently we run on both Solaris and Linux). These include things like addressable memory, number of open file descriptors, network sockets and bandwidth, and many others. We believe expanding to a lot more than 100 million pages would greatly increase the complexity of our system.

9.2 Scalability of Centralized Indexing Architectures

As the capabilities of computers increase, it becomes possible to index a very large amount of text for a reasonable cost. Of course, other more bandwidth intensive media such as video is likely to become more pervasive. But, because the cost of production of text is low compared to media like video, text is likely to remain very pervasive. Also, it is likely that soon we will have speech recognition that does a reasonable job converting speech into text, expanding the amount of text available. All of this provides amazing possibilities for centralized indexing. Here is an illustrative example. We assume we want to index everything everyone in the US has written for a year. We assume that there are 250 million people in the US and they write an average of 10k per day. That works out to be about 850 terabytes. Also assume that indexing a terabyte can be done now for a reasonable cost. We also assume that the indexing methods used over the text are linear, or nearly linear in their complexity. Given all these assumptions we can compute how long it would take before we could index our 850 terabytes for a reasonable cost assuming certain growth factors. Moore's Law was defined in 1965 as a doubling every 18 months in processor power. It has held remarkably true, not just for processors, but for other important system parameters such as disk as well. If we assume that Moore's law holds for the future, we need only 10 more doublings, or 15 years to reach our goal of indexing everything everyone in the US has written for a year for a price that a small company could afford. Of course, hardware experts are somewhat concerned Moore's Law may not continue to hold for the next 15 years, but there are certainly a lot of interesting centralized applications even if we only get part of the way to our hypothetical example.

Of course a distributed systems like G*l*oss [Gravano 94] or Harvest will often be the most efficient and elegant technical solution for indexing, but it seems difficult to convince the world to use these systems because of the high administration costs of setting up large numbers of installations. Of course, it is quite likely that reducing the administration cost drastically is possible. If that happens, and everyone starts running a distributed indexing system, searching would certainly improve drastically.

Because humans can only type or speak a finite amount, and as computers continue improving, text indexing will scale even better than it does now. Of course there could be an infinite amount of machine generated content, but just indexing huge amounts of human generated content seems tremendously useful. So we are optimistic that our centralized web search engine architecture will improve in its ability to cover the pertinent text information over time and that there is a bright future for search.